Universidade Federal de Santa Catarina
Programa de Pós-graduação em Engenharia de Produção

Rivalino Matias Júnior

ENVELHECIMENTO DE SOFTWARE UTILIZANDO ENSAIOS DE VIDA ACELERADOS QUANTITATIVOS

Tese de Doutorado

Florianópolis
2006

Rivalino Matias Júnior

ENVELHECIMENTO DE SOFTWARE UTILIZANDO ENSAIOS DE VIDA ACELERADOS QUANTITATIVOS

Tese apresentada ao programa de Pós-graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de doutor em Engenharia de Produção.

Orientador: Paulo José de Freitas Filho, Dr.

Florianópolis
2006

Rivalino Matias Júnior

ENVELHECIMENTO DE SOFTWARE UTILIZANDO ENSAIOS DE VIDA ACELERADOS QUANTITATIVOS

Esta tese foi julgada adequada e aprovada para a obtenção do título de **Doutor em Engenharia de Produção** no **Programa de Pós-Graduação em Engenharia de Produção** da Universidade Federal de Santa Catarina

Florianópolis, 24 de Agosto de 2006.

Prof. Antônio Sérgio Coelho, Dr.
Coordenador do Curso

BANCA EXAMINADORA

Prof. Paulo José de Freitas Filho, Dr. Orientador	Prof. João Bosco da Mota Alves, Dr. Moderador
Prof. Celso Maciel da Costa, Dr. Examinador Externo	Prof. Murilo Silva de Camargo, Dr. Examinador Externo
Prof. Acires Dias, Dr. Membro	Profa. Elizabeth Sueli Specialski, Dra. Membro
Prof. Mario Dantas, Ph.D. Membro	Prof. Pedro Alberto Barbetta, Dr. Membro

AGRADECIMENTOS

Agradeço a Deus pelas oportunidades e sua proteção constante.

Aos meus pais pelo apoio, confiança e incentivo aos estudos durante toda minha vida.

À minha esposa Luciana Bortolus Matias que sempre esteve presente me incentivando e colaborando em todas as etapas desta jornada.

Ao meu orientador Prof. Paulo José de Freitas Filho pela amizade, confiança, parceria, respeito, conhecimento e sabedoria ao me orientar durante este projeto.

À minha grande amiga e sempre orientadora Profa. Elizabeth Specialski pela amizade e incentivo sempre presentes.

Ao Professor e amigo Pedro Alberto Barbetta pelas importantes e significativas contribuições que foram essenciais para concluir este trabalho com êxito.

Aos vários colegas que tive a oportunidade de conviver durante as disciplinas do curso, em especial à minha grande amiga Vera do Carmo Comparsi de Vargas quem me ensinou muito sobre estatística, o que foi de fundamental importância para a condução deste estudo.

Aos membros da banca avaliadora, profissionais de grande competência que tive a honra de contar com suas valiosas contribuições para enriquecer este trabalho.

Ao meu grande amigo Prof. Carlos Antunes pela sua amizade, sabedoria e incentivo nos primeiros passos, o que foi muito importante para que eu chegasse até aqui.

À Universidade Federal de Santa Catarina, em especial às equipes do PPGEP, NPD, PLAB (INE) e PGCC (Vera Lúcia Sodré Teixeira), os quais sempre estiveram à disposição com muita dedicação e amizade.

Aos autores e pesquisadores que contribuíram com seus estudos prévios servindo de base para esta pesquisa.

RESUMO

Este trabalho apresenta uma abordagem sistematizada para acelerar o tempo de vida de sistemas que são acometidos pelos efeitos do envelhecimento de software. Estudos de confiabilidade voltados para estes sistemas necessitam realizar a observação dos tempos de falhas causadas pelo envelhecimento de software, o que exige experimentos de longa duração. Esta exigência cria diversas restrições, principalmente quando o tempo de experimentação implica em prazos e custos proibitivos para o estudo. Neste sentido, este trabalho apresenta uma proposta para acelerar a vida de sistemas que falham por envelhecimento de software, reduzindo o tempo de experimentação necessário para observar as suas falhas, o que reduz os prazos e custos das pesquisas nesta área. A fundamentação teórica deste estudo contou com um arcabouço conceitual envolvendo as áreas de *dependabilidade* computacional, engenharia de confiabilidade, projeto de experimentos, ensaios de vida acelerados e o estudo da fenomenologia do envelhecimento de software. A técnica de aceleração adotada foi a de ensaios de degradação acelerados, a qual tem sido largamente utilizada em diversas áreas da indústria, mas até o momento não tinha sido usada em estudos envolvendo produtos de software. A elaboração dos meios que permitiram aplicar esta técnica no âmbito da engenharia de software experimental, abordando especialmente o problema do envelhecimento de software, é a principal contribuição desta pesquisa. Em conjunto com a fundamentação teórica foi possível avaliar a aplicabilidade do método proposto a partir de um estudo de caso real, envolvendo a aceleração do envelhecimento de um software servidor web. Dentre os principais resultados obtidos no estudo experimental, destaca-se a identificação dos tratamentos que mais contribuíram para o envelhecimento do software servidor web. A partir destes tratamentos foi possível definir o padrão de carga de trabalho que mais influenciou no envelhecimento do servidor web analisado, sendo que o tipo e tamanho de páginas requisitadas foram os dois fatores mais significativos. Outro resultado importante diz respeito à verificação de que a variação na taxa de requisições do servidor web não influenciou o seu envelhecimento. Com relação à redução no período de experimentação, o método proposto apresentou o menor tempo em comparação aos valores previamente reportados na literatura para experimentos similares, tendo sido 3,18 vezes inferior ao menor tempo encontrado. Em termos de MTBF estimado, com e sem a aceleração do envelhecimento, obteve-se uma redução de aproximadamente 687 vezes no tempo de experimentação aplicando-se o método proposto.

ABSTRACT

This research work presents a systematic approach to accelerate the lifetime of systems that fail due to the software aging effects. Reliability engineering studies applied to systems that require the observation of time to failures caused by software aging normally require a long observation period. This requirement introduces several practical constraints, mainly when the experiment duration demands prohibitive time and cost. The present work shows a proposal to accelerate the lifetime of systems that fail due to software aging, reducing the experimentation time to observe their failures, which means smaller time and costs for research works in this area. The theoretical fundamentals used by the proposed method were based on concepts of the following areas: computing dependability, reliability engineering, design of experiments, accelerated life tests and the software aging phenomenology. The lifetime acceleration technique adopted was the quantitative accelerated degradation test. This technique is largely used in several industry areas, however until the moment it hadn't been used in the software engineering field. The specification of means that allowed applying this technique to the experimental software engineering area, especially to approach the software aging problem, it is considered the main contribution of this research work. Also, it was possible to evaluate the applicability of the proposed method in a case study related to the software aging acceleration of a real web server. An important result was the identification of treatments that contributed to the web server aging. Based on these treatments was possible to define a workload standard that most influenced the aging effects on the web server analyzed, where the page size and page type were two significant factors. Another important result of this case study is regarding the request rate variability, that hadn't influence on the aging of the investigated web server software. Regarding the reduction of the experimentation period, the proposed method showed a shorter duration than values from similar experiments previously published, being 3.18 times less than the shorter experimentation time found in the literature. In terms of MTBF estimates, obtained with and without the aging acceleration, it was possible to achieve a reduction of approximately 687 times of the experimentation time using the proposed method.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ADT	Accelerated Degradation Tests
ALT	Accelerated Life Tests
ANOVA	Analysis of Variance
ARMA	Autoregressive Moving Average
AST	Accelerated Stress Testing
AT&T	American Telephone and Telegraph
CDF	Cumulative Distribution Function
CFE	Código Freqüentemente Executado
CGI	Common Gateway Interface
CNFE	Código Freqüentemente não Executado
CPU	Central Processing Unit
CTMC	Continuous-Time Markov Chain Model
DOE	Design Of Experiment
DT	Degradation Test
E/S	Entrada e/ou Saída
ESS	Environment Stress Screening
FCFS	First-Come-First-Served
FOSS	Free/Open Source Software
FRE	Falta(s) Relacionada(s) ao Envelhecimento
FTP	File Transfer Protocol
GAO/IMTEC	United States General Accounting Office / Information Management and Technology Division
GOF	Goodness-of-Fit
HALT	High Accelerated Life Test
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machine
IC (x%)	Intervalo de x% de Confiança
IEEE	Institute of Electrical and Electronics Engineers
IID	Independente e Identicamente Distribuído
IPL	Inverse Power Law

ISP	Internet Service Provider
JPL	Jet Propulsion Laboratory
KB	Kilobytes
K-S	Teste de aderência Kolmogorov-Smirnov
LI	Limite Inferior de um Intervalo de Confiança
Lk	Likelihood Value
LOC	Lines of Code
LS	Limite Superior de um Intervalo de Confiança
LSE	Least Squares Estimation
MBPS	Megabits per Second
MIB	Management Information Base
MIPS	Milhões de Instruções por Segundo
MLE	Maximum Likelihood Estimation
MPM	Multiprocessing Modules
MRM	Markov Reward Model
MRSPN	Markov Regenerative Stochastic Petri Net
MSET	Multivariate State Estimation Technique
MTBF	Mean Time Between Failure
MTTF	Mean Time to Failure
N/A	Não Aplicável
NASA	National Aeronautics and Space Administration
NFS	Network File System
NIC	Network Interface Card
NU	O Nível de Uso do SST em termos de Carga de Trabalho
PDF	Probability Density Function
PFM	Proactive Fault Management
PMBOK	Project Management Body of Knowledge Guide
QoS	Quality of Service
RAM	Random Access Memory
RCA	Root Cause Analysis
RPC	Remote Procedure Call
SGBD	Sistema Gerenciador de Base de Dados
SMP	Semi-Markov Process

SNMP	Simple Network Management Protocol
SO	Sistema Operacional de Computador
SPRT	Sequential Probability Ratio Test
SQT	Soma dos Quadrados Totais
SQx	Soma dos Quadrados de um Fator (ex. SQA) ou de suas Interações (ex. SQAB)
SRN	Stochastic Reward Net
SSH	Secure Shell Protocol
SST	Sistema/produto Sob Teste
TCP	Transport Control Protocol
TEA	Teste(s) de Envelhecimento Acelerado
TEC	Tempo entre Chegadas de Requisições para o SST
TELNET	Protocolo de Terminal Virtual
TI	Tecnologia da Informação
TMP	Tamanho Médio de Página HTML
TP	Tipo de Página HTML
TR	Taxa de Requisições HTTP
TTF	Time To Failure
TTPF	Time To Pseudo-Failure
UFSC	Universidade Federal de Santa Catarina
URL	Universal Reference Location
WEB	World Wide Web

NOTAÇÃO

2^k, 2^kr e 2^{k-p}r	tipos de projetos de experimentos (seção 5.3.1)
n_{TEA}	número de replicações do TEA (seções 6.4.2.2 e A.2)
α	nível de significância do teste *F* que acompanha a ANOVA (seção 5.3.2) / nível de significância para um teste K-S (seção 5.5.1)
β	parâmetro de forma (*shape*) da distribuição Weibull (seção 4.2.5.2)
β_0, β_1	coeficientes do modelo IPL linearizado (seção 4.2.5.1.1) / coeficientes de regressão (seção 4.2.5.2)
$\beta=(\beta_{i1},\ldots,\beta_{ik})'$	vetor de coeficientes do modelo para o caminho de degradação D_{ij} (seção 4.2.6.1)
$D(t)$, D_{ij}	caminho(s) de degradação (seção 4.2.6.1)
D_f	nível crítico de degradação de (γ) em um ADT/TEA (seção 4.3.1)
D_t	duração do ADT/TEA que define o limiar de censura por tempo (seção 4.3.1)
$ep(\cdot)$	erro padrão de um determinado efeito (seção 5.3.1)
$ef(\cdot)$	efeito de um determinado fator (seção 5.3.2.1)
E_0	erro amostral tolerado para o cálculo do número de replicações do DOE (seção 5.3.1)
ε	componente de erro aleatório do modelo de relacionamento vida-estresse (seção 4.2.5.2)
f(estresse)	forma funcional do relacionamento vida-estresse (seções 4.2.5.1, 4.2.5.2)
$f(t)$	função densidade de probabilidade (PDF) dos tempos de falha de um determinado SST (seção 4.2.5.3)
$f(t,V)$	função densidade de probabilidade (PDF) dos tempos de falha para um determinado nível de estresse (seção 4.2.5.3)
$F(t)$	função de distribuição acumulada (CDF) ou função de não-confiabilidade (*unreliability*) (seção 4.2.6.2)
F_c	valor crítico tabulado para um teste com distribuição *F de Snedecor* (seção 6.4.2)
F_E	fator de envelhecimento (seção 5.2)
χ^2	teste de aderência Qui-quadrado (seção 5.5.1)
ρ	coeficiente de correlação linear de *Pearson* (seção 6.4.2.2)
λ	constante da taxa de falha (seções 2.3 e 3.3.1) / parâmetro da distribuição exponencial (seção 4.2.5.3)
$\lambda(t,V)$	função taxa de falha para um dado nível de estresse (seção 4.2.5.2)

μ	parâmetro de localização da CDF de uma função distribuição da família localização-escala (seção 4.2.5.2)
Log_e ou $\ln(\cdot)$	logaritmo natural (seção 5.6)
$N(\mu,\sigma)$	significa estar distribuído de acordo com uma distribuição de probabilidades normal com média μ e desvio padrão σ (seção 4.2.6.1)
N_a	nível superior (alto) de um plano experimental no ADT/TEA (seção 4.3.2.2)
N_b	nível inferior (baixo) de um plano experimental no ADT/TEA (seção 4.3.2.2)
N_i	nível intermediário de um plano experimental no ADT/TEA (seção 4.3.2.3)
Valor *p*	probabilidade de significância para a estatística de um teste de hipótese (apêndice C)
$Pr(\cdot)$	probabilidade do evento $(\cdot)$ ocorrer (seção 4.2.5.2)
$R(t,V)$	função confiabilidade para um dado nível de estresse (seção 4.2.5.3)
R^2	coeficiente de determinação (seção 6.4.2.2)
R1, R2, ...	identificador da replicação do DOE (seção 5.3.2.1) / identificador da replicação do TEA (apêndice B)
s^2	variância da amostra piloto do DOE (seção 6.4.1) / variância da amostra piloto do TEA (seção 6.4.2.2 e apêndice A)
S_0	estado robusto de um processo de aplicação/software (seção 3.3.1)
S_i	identificador do nível *i* de estresse (ex S1, S2, S3) (seção 5.5.1)
S_P	estado de falha provável (*failure probable state)* (seção 3.3.1)
S_R	estado de reparação preventiva (seção 3.3.1)
t_{ij}	o *j-ésimo* tempo de inspeção (observação) da medida de degradação (γ) da *i-ésima* unidade de teste (seção 4.2.6.1)
$t_{\alpha/2,gl}$	valor tabulado da distribuição *t de Student* para $\alpha/2$ *e gl* graus de liberdade do erro (seção 5.3.1)
V	nível de estresse (seção 4.2.5.1.1)
gl ou v	graus de liberdade (seções 5.3.1, 6.4.1.2 e apêndice A)
y	variável resposta do DOE (seção 5.3.1)
$\overline{y}_{i.}$ e $\overline{y}_{..}$	média das observações do *i-ésimo* tratamento (seção 5.3.2.1) e média global do experimento (seção 5.3.2.1)
γ_{ij}	a medida de degradação observada na unidade *i* no tempo t_{ij} (seção 4.2.6.1)
z	valor tabulado da distribuição Normal padronizada (apêndice A)
δ	número total de tratamentos do DOE (seção 5.3.1)
η	parâmetro de escala da distribuição Weibull (seção 4.2.5.2)
π_i	proporção de alocação de unidades de teste no *i-ésimo* nível de estresse (seção 4.3.2.1)

σ	parâmetro de escala da CDF de uma função distribuição da família localização-escala (seção 4.2.5.2)
$\tau(\cdot)$	tempo de vida médio de um SST para um dado nível de estresse (seção 4.2.5.1.1)
Φ	CDF de uma função distribuição de probabilidades da família localização-escala (seção 4.2.5.2)

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

Atualmente o software permeia praticamente todos os aspectos da sociedade moderna. Sua aplicação se estende por áreas governamentais, indústria, comércio e serviços, enfim praticamente todas as demais áreas que, direta ou indiretamente, têm influência de alguma ordem na vida das pessoas. Sem o software muitas conveniências da sociedade atual seriam virtualmente impossíveis (TORRES-POMALES, 2000).

Não obstante à grande importância do software atualmente, em muitos casos controlando recursos e/ou sistemas[1] de missão crítica, está bem estabelecido tanto pela academia quanto pela indústria que a crescente complexidade dos sistemas de software têm tornado cada vez mais difícil se atingir o objetivo de construir software sem erros. Não é difícil imaginar que a crescente dependência e complexidade do software, transformam as falhas de software em eventos cujo impacto pode atingir níveis de importância bastante elevados, podendo, em alguns casos, tomar proporções consideradas catastróficas. Existem diversos casos envolvendo falhas de software, cujos impactos foram considerados de grande magnitude, dentre os quais serão apresentados alguns exemplos a seguir.

De 1985 até 1987 seis pessoas foram inadequadamente expostas a níveis elevados de radiação durante seus tratamentos de câncer utilizando o equipamento médico de terapia radioativa chamado Therac-25. Três destes pacientes morreram devido a esta superexposição. Após a análise e detecção do funcionamento incorreto do equipamento, verificou-se que a causa raiz do problema havia sido uma falha na rotina de entrada de dados do software de controle do Therac-25 (LEVERSON; TURNER, 1993). Em 4 de junho de 1996, o foguete não tripulado Ariane 5, construído pela agência espacial européia, explodiu quarenta segundos após o seu lançamento. O custo do desenvolvimento do Ariane-5, que levou uma década para ser construído, foi de US$ 7 bilhões e o equipamento destruído teve valor estimado em US$ 500 milhões. Após a investigação das causas do acidente, concluiu-se que um erro no software do sistema de referência inercial do foguete foi responsável pela falha que causou a

[1] Neste trabalho o termo sistema representa uma entidade que interage com outras entidades (ex. outros sistemas) incluindo hardware, software, humanos e o mundo físico.

explosão do Ariane-5. Especificamente, o erro se localizava em uma conversão inadequada de um valor real de 64 bits, maior que 32.768[2], para uma variável do tipo inteiro sinalizado de 16 bits, o que acarretou a falha e explosão do foguete (NEUMANN, 1995). Outro caso documentado ocorreu em 25 de fevereiro de 1991, durante a primeira guerra do Golfo. Uma bateria antiaérea Patriot, do exército dos Estados Unidos, falhou ao interceptar mísseis *Sculd* lançados pelo Iraque contra um acampamento do exército norte americano. Esta falha provocou a morte de vinte e oito soldados dos Estados Unidos. O relatório GAO/IMTEC-92-26 (1992), intitulado "*Patriot Missile Software Problem*", descreveu como causa raiz que levou à falha da bateria, um erro do software de controle do sistema Patriot. Outros casos relacionados à falhas catastróficas causadas por software podem ser encontrados em Dershowitz (2005).

Para Brooks (1987), a dificuldade em se atingir a correta construção de um software origina-se de quatro atributos: complexidade, conformidade, volatilidade e invisibilidade. Brooks argumenta que a complexidade é provocada pela grande quantidade de estados presentes em um software e as interações não lineares entre estes estados. Devido à sua natureza volátil, o software, dentre todos os componentes de um sistema, é aquele do qual se exige maior conformidade e maleabilidade entre modificações no sistema. Mesmo quando o software não é o alvo direto de uma mudança nas funcionalidades do sistema, os componentes de software do sistema, muitas vezes, serão alterados para acomodar as novas funcionalidades. Como atributo final, Brooks salienta a invisibilidade do software. A representação do software através de linguagens de programação se torna limitada a uma representação estática, a qual não permite capturar diversas condições e, principalmente, interações existentes em tempo de execução (representação dinâmica).

A preocupação com a crescente complexidade do software originou, há mais de quarenta anos, a disciplina de engenharia de software, que busca o estabelecimento e aplicação de práticas sistematizadas para a construção e avaliação de produtos de software (PRESSMAN, 2005). Segundo IEEE (1990), engenharia de software é a aplicação de um enfoque sistemático, organizado e científico, para o desenvolvimento, operação e manutenção de software. Enfatiza-se que, de acordo com esta definição, a engenharia de software contempla não somente a etapa de desenvolvimento e manutenção, mas também a execução do software, ou seja, o seu estado operacional (dinâmico).

[2] Maior valor inteiro (sinalizado) armazenável em uma posição de memória de 16 bits.

Devido ao importante papel que o software tem ocupado, principalmente naqueles sistemas de natureza crítica, um importante conceito dentro da engenharia de software tem sido o de *dependabilidade*[3]. A *dependabilidade* é um dos aspectos da qualidade que tem relação com o tempo e uso de um produto de software. Em Avižienis *et al.* (2004), o conceito de *dependabilidade* é abordado como uma das cinco propriedades fundamentais de um sistema computacional, as quais são: funcionalidade, desempenho, *dependabilidade***,** segurança e custo.

Por *dependabilidade* de um sistema computacional entende-se a habilidade deste sistema em fornecer um serviço no qual seu usuário[4] pode, justificadamente, ter confiança no seu funcionamento. Este conceito está diretamente relacionado ao conceito de qualidade de serviço (QoS), ou seja, a percepção por parte do usuário do sistema a respeito do seu comportamento (serviço prestado) (VERÍSSIMO; LEMOS, 1989). Segundo Avižienis *et al.* (2004), a definição de *dependabilidade* é composta por três elementos, os quais são: impedimentos ou ameaças (*threats*), meios (*means*) e medidas (*attributes*). Os impedimentos à *dependabilidade* são circunstâncias indesejáveis, não esperadas, que levam o sistema a não fornecer corretamente o serviço requerido pelo seu usuário. Os meios, por sua vez, são os métodos, ferramentas e soluções que possibilitam ao sistema ter capacidade para fornecer o serviço no qual se possa depositar confiança, garantindo que esta confiança seja justificada. Os atributos têm por objetivo oferecer medidas de *dependabilidade*, as quais permitem justificar a confiança depositada na capacidade do sistema em prover um dado serviço.

Como se pode observar, a definição de *dependabilidade* integra diversos conceitos, os quais têm sido desenvolvidos nas últimas três décadas (AVIŽIENIS *et al.*, 2004). A seguir serão apresentados os relacionamentos entre este trabalho e os conceitos (ramificações) ilustrados na figura 1.1.

---

[3] O termo *dependabilidade* é definido na norma NBR ISO 9000-4 (ABNT, 1993). Alguns autores têm utilizado as expressões "confiança no funcionamento" e "computação confiável" como equivalentes aos termos em inglês *dependability* e *dependability computing*. Sugestões de versões em português, dos diversos termos relacionados a esta área, são descritas na terminologia proposta em Veríssimo e Lemos (1989). Neste texto será preferencialmente utilizado o termo *dependabilidade*, podendo em alguns casos ser substituído pelas alternativas já citadas, de acordo com a conveniência e contextualização de suas ocorrências.

[4] Neste contexto, um usuário é outro sistema (humano ou físico) que interage com o sistema em questão.

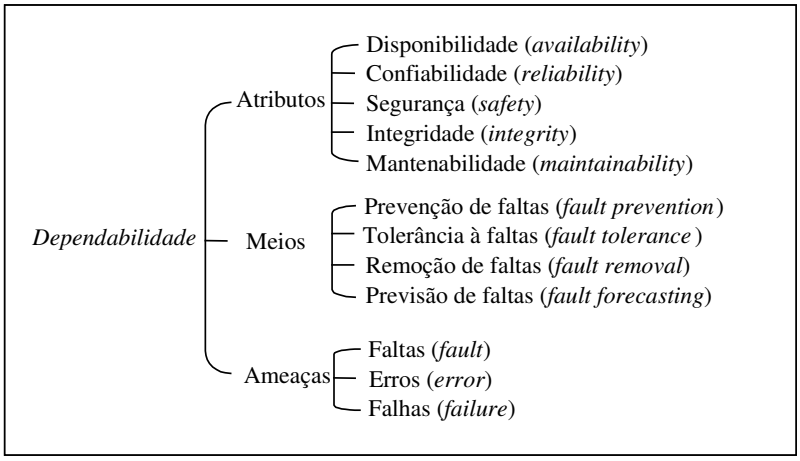

Figura 1.1: Árvore de *dependabilidade*
Fonte: Traduzido e adaptado de Avižienis *et al.* (2004)

No que tange às ameaças ou impedimentos à *dependabilidade*, este trabalho tem como escopo abordar o problema do envelhecimento de software. Sucintamente, o envelhecimento de software pode ser entendido como o acúmulo de erros (ocorrência de faltas internas) no software durante sua execução, levando a uma degradação progressiva do seu estado interno, bem como a exaustão de recursos essenciais[5] para seu correto funcionamento, o que causa na maioria das vezes a falha do sistema. Um exemplo que representa bem o problema do envelhecimento de software é o vazamento de memória (*memory leak*). No capítulo 3 o problema do envelhecimento de software, juntamente com alguns casos envolvendo vazamento de memória, serão apresentados em detalhes.

Com relação aos atributos ou medidas, esta pesquisa tem como abordagem central a confiabilidade, estando diretamente relacionada com os aspectos da engenharia de confiabilidade operacional de software. Nesta linha, o trabalho propõe uma abordagem sistematizada que visa reduzir o tempo necessário para a obtenção de dados de vida (tempos de falha) de sistemas que falham por envelhecimento de software. A partir da distribuição dos tempos de falha destes sistemas é possível obter as suas estimativas de confiabilidade operacional, como por exemplo, MTTF (*Mean Time to Failure*), MTBF (*Mean Time Between Failure*), taxa de falhas, dentre outras. Com base nestas métricas, os meios adequados para se atingir a *dependabilidade* poderão ser selecionados e aplicados.

Dentre os meios listados na figura 1.1, a área de tolerância à faltas tem sido a de maior aplicação em pesquisas envolvendo o envelhecimento de software, em especial no

[5] Recursos necessários para o funcionamento do software, os quais são provenientes do sistema operacional. São exemplos: memória principal, descritores de arquivos, portas de conexão, outros.

desenvolvimento de mecanismos de mitigação do envelhecimento chamados de rejuvenescimento de software (HUANG *et al.*, 1995; XIE; HONG; TRIVEDI, 2005).

A seguir serão apresentadas justificativas que destacam a importância deste trabalho no contexto descrito nesta seção.

1.2. RELEVÂNCIA DO TRABALHO

Como já citado anteriormente, os diversos aspectos da *dependabilidade* de software se tornam cada vez mais importantes na medida que a utilização extensiva do software se faz presente em diversas áreas da sociedade. Um problema de pesquisa seriamente investigado no campo da *dependabilidade* computacional tem sido o envelhecimento de software, cujos efeitos incidem direta e negativamente sobre os atributos de confiabilidade e disponibilidade de sistemas que, de alguma forma, possuem em suas partes componentes de software. Maiores detalhes sobre o envelhecimento de software serão abordados no capítulo 3.

Um caso real e de grandes proporções envolvendo falha causada por envelhecimento de software, foi citado na seção 1.1, referente ao sistema militar Patriot. A partir da análise do equipamento após a sua falha, se comprovou que o acúmulo de erros de precisão numérica, após 8 horas de execução ininterrupta do seu software de controle, permitiu reproduzir a falha. O software de controle da bateria, que falhou na interceptação do míssil *Scud*, já estava em funcionamento há aproximadamente 100 horas quando ocorreu o incidente. A solução paliativa para o caso em questão, ainda durante a guerra do Golfo, foi a reinicialização (*reboot*) programada do software em períodos de 8 horas, o que permitia a operação segura da bateria (MARSHALL, 1992). Ressalta-se que durante o intervalo de tempo da reinicialização do equipamento, a região protegida pelo mesmo se encontrava vulnerável.

Exemplos como o descrito anteriormente têm exigido da indústria investimentos em pesquisas para tornar os produtos cada vez mais tolerantes a problemas desta natureza. Os esforços da indústria nesta área ficam evidentes, a partir dos diversos trabalhos publicados por institutos de pesquisa e organizações tais como IBM (CASTELLI *et al.*, 2001), AT&T (AVRITZER; WEYUKER, 1997), Microsoft (TECHNET, 2001), NASA (TAI; ALKALAI, 1998), dentre outros. Além da produção científica amplamente divulgada, empresas como IBM têm desenvolvido tecnologias visando ampliar a *dependabilidade* de seus sistemas comerciais que, de alguma forma, estão vulneráveis ao problema do envelhecimento de software (YANG; MELENOVSKY, 2004; CASTELLI *et al.*, 2001). Também, constatou-se que nos Estados Unidos, desde 1996 têm sido registradas patentes de software à cerca de

tecnologias voltadas para a solução do problema do envelhecimento de software (FULTON *et al.*, 1996; HARPER; HUNTER, 1999, 2004; HARPER *et al.*, 1999), o que demonstra a importância dada pela indústria com relação a esta área de pesquisa.

A partir de uma extensa revisão da literatura, verificou-se que a maior parte dos esforços neste campo tem se concentrado na caracterização do fenômeno do envelhecimento de software, bem como em propostas de modelos analíticos para a predição dos tempos de falhas causadas pelo envelhecimento. O desenvolvimento de meios/mecanismos para a mitigação dos efeitos do envelhecimento, também tem sido alvo de uma parcela menor de trabalhos. Estas pesquisas, em grande parte, convergem no sentido de proporcionar maior confiança no funcionamento de sistemas computacionais que estão susceptíveis aos efeitos do envelhecimento de software. Uma tendência nos trabalhos publicados recentemente tem sido a adoção de um enfoque integrado, envolvendo medições do envelhecimento a partir de experimentação ou observação, com a posterior construção de modelos analíticos para representar o fenômeno do envelhecimento de software. No capitulo 3 esta tendência será evidenciada a partir da apresentação do histórico de evolução dos trabalhos nesta área.

Durante a revisão da literatura, verificou-se que nos trabalhos puramente experimentais, ou com abordagem mista (analítica e experimental), os experimentos realizados não foram conduzidos até a ocorrência de falhas causadas pelo envelhecimento. A partir dos dados reportados nestes estudos, verificou-se que o comportamento do envelhecimento que foi caracterizado, tipicamente exigiria um período longo de execução até a ocorrência de falhas causadas pelos efeitos do envelhecimento. Este cenário fica evidente naquelas publicações que reportam o tempo de duração da sua etapa experimental, como é o caso dos trabalhos que serão comentados a seguir.

Em Li, Vaidyanathan e Trivedi (2002), três experimentos foram conduzidos totalizando 46 dias de observação ininterrupta de um software servidor web, cujo propósito foi de se verificar a existência dos efeitos do envelhecimento neste sistema. Salienta-se que os autores não observaram o sistema até sua falha, o que necessitaria de um tempo maior de experimentação. Além disso, no caso da necessidade de replicação dos experimentos, o que normalmente ocorre, o tempo total da fase experimental seria substancialmente ampliado. Já em Avritzer e Weyuker (1997), relatou-se a necessidade de observação de sistemas de telecomunicações por várias semanas, a fim de detectar indícios de degradação em seus recursos como conseqüência do envelhecimento de software. Esta afirmação é compatível com os resultados reportados em Huang *et al.* (1995), que também desenvolveu pesquisa em envelhecimento de software voltada para sistemas de telecomunicações. Huang *et al.*, durante

duas semanas, monitoraram ininterruptamente os sintomas de envelhecimento de software no sistema de coleta de dados de bilhetagem (BILL-DATS II© Collector) da companhia AT&T. Nos experimentos realizados por Trivedi, Vaidyanathan e Goseva-Popstojanova (2000), foram realizadas coletas, em intervalos de 15 minutos, durante um período de 53 dias, a fim de se estudar os efeitos do envelhecimento de software em estações de trabalho (UNIX). Em outra pesquisa similar, Vaidyanathan e Trivedi (2001b) adotam um período superior a três meses para o monitoramento também de estações de trabalho, objetivando a detecção do envelhecimento de software neste tipo de ambiente. As duas últimas referências apresentam períodos compatíveis com os resultados descritos em Matias Jr. *et al.* (2005), que a partir da simulação do envelhecimento de cinqüenta servidores web, obteve um tempo médio de falhas por envelhecimento de 1.780 horas. Todos os trabalhos citados, bem como demais pesquisas envolvendo o envelhecimento de software, serão descritos em detalhes no capítulo 3.

Como fica evidente a partir dos exemplos citados anteriormente, a obtenção de dados de vida de sistemas que falham por envelhecimento de software exige consideráveis períodos de experimentação, o que tem relação direta com o custo e a viabilidade da pesquisa. A coleta de tempos de falha de sistemas que falham por envelhecimento de software, a fim de se conduzir uma posterior análise de confiabilidade, pode se tornar muito onerosa principalmente em se tratando de componentes de software com níveis elevados de confiabilidade, como por exemplo, em sistemas de alta disponibilidade (ex. *clusters*). Neste sentido, alguns trabalhos têm sugerido a utilização de modelos de previsão para estimar os tempos de falha a partir de uma amostra de dados coletados com respeito aos efeitos do envelhecimento do sistema. Apesar de ser uma abordagem apropriada para alguns casos, o período de experimentação necessário para se capturar uma amostra significativa do padrão de comportamento dos efeitos do envelhecimento, a fim de se realizar a predição dos tempos de falha do sistema, pode ser muito elevado se considerando a execução do software em condições normais de operação. Este cenário é discutido em Ehrlich *et al.* (1998).

A contribuição deste trabalho aborda esta problemática, ou seja, de que experimentos realizados para observar falhas por envelhecimento de software normalmente exigem longos períodos de observação, o que torna onerosa a coleta de dados para posteriormente se analisar a confiabilidade destes sistemas. Neste sentido, a presente pesquisa propõe um método para acelerar a manifestação dos efeitos do envelhecimento de software, de forma que as falhas por envelhecimento possam ser observadas em um tempo inferior àquele necessário durante a

operação normal do sistema. A partir de então, estima-se a distribuição de vida[6] do sistema, a qual será usada para a sua posterior análise de confiabilidade. Para atingir este objetivo, o método proposto adota a técnica de ensaios acelerados (NELSON, 2004) no âmbito da engenharia de software. O propósito deste enfoque é reduzir os custos (tempo) da experimentação, durante a obtenção dos dados de vida (tempos de falha) de componentes de software que falham por envelhecimento e que possuem como característica uma grande longevidade operacional.

De fato, a necessidade de acelerar a obtenção de dados de vida em software não é diferente de outras áreas da indústria, principalmente quando se testam espécimes[7] projetados para terem grande durabilidade ou quando o fenômeno que causa a falha se manifesta somente após um longo período de observação. Nas demais áreas da indústria, a utilização de ensaios de vida acelerados (ALT – *Accelerated Life Tests*) tem sido uma prática cada vez mais adotada em estudos de confiabilidade de sistemas, haja vista a redução significativa dos tempos e, conseqüentemente, dos custos (NELSON, 2004). A principal diferença em se adotar esta técnica para a aceleração de falhas em software, em relação às demais áreas, reside nos modelos de aceleração de vida (ou de degradação) necessários para a sua implementação. Atualmente, a utilização de ensaios acelerados aplicados a componentes de software é uma área pouco explorada pela academia e indústria. Esta afirmação está baseada em uma extensa revisão da literatura nos campos da engenharia de software experimental, engenharia de confiabilidade de software, *dependabilidade* computacional, tolerância à faltas, dentre outras áreas correlatas.

Apesar da flexibilidade apresentada pelas técnicas de ALT, segundo Meeker, Escobar e Lu (1998) existem casos específicos em que mesmo utilizando ALT a obtenção dos tempos de falha exige longos períodos de experimentação, como, por exemplo, nos testes de componentes altamente confiáveis, construídos com elevados requisitos de confiabilidade e disponibilidade (ex. componentes de satélites). De acordo com os mesmos autores, nestas situações uma solução é abordar o problema através de ensaios de degradação acelerados. Os ensaios de degradação acelerados (ADT – *Accelerated Degradation Tests*) (MEEKER; ESCOBAR; LU, 1998) são uma alternativa e, em alguns casos, um complemento da técnica de ensaios de vida acelerados. A utilização tanto de ALT quanto de ADT, no contexto da

[6] O termo "distribuição de vida", neste trabalho, se refere à distribuição dos tempos de falha do sistema analisado.

[7] Objeto sendo testado podendo ser de qualquer natureza (ex. componente eletrônico, peça mecânica, substância química, outros). Neste trabalho o termo se refere a componentes de software.

engenharia de software, requer a introdução de conceitos que mapeiem os mecanismos de aceleração/degradação, presentes nas demais áreas onde tradicionalmente estas técnicas têm sido utilizadas, com aqueles existentes no âmbito do software. Como será discutido nos capítulos 4 e 5, a ausência de trabalhos envolvendo ALT/ADT no campo da engenharia de software exige que a fundamentação de aspectos importantes deste trabalho seja com base em experimentos estatisticamente planejados. Os resultados desta etapa experimental servirão para avaliar os modelos de aceleração utilizados pelo método proposto.

A partir da argumentação anterior, entende-se que este trabalho se mostra relevante para o campo da pesquisa experimental em *dependabilidade* computacional, especificamente no que tange aos seus aspectos de confiabilidade e tolerância à faltas provocadas por envelhecimento de software.

1.3. OBJETIVOS DA PESQUISA

A seguir serão apresentados os objetivos (geral e específicos) do trabalho.

1.3.1. Objetivo geral

Propor uma abordagem sistematizada para acelerar as falhas de sistemas causadas por envelhecimento de software.

1.3.2. Objetivos específicos

- Avaliar experimentalmente o conceito de fator de envelhecimento proposto neste trabalho;
- Testar a adequação do modelo de potência inversa (IPL – *Inverse Power Law*) para analisar os dados de vida (tempos de falha) obtidos com a aceleração do envelhecimento de software;
- Analisar quantitativamente a acurácia de um modelo de aceleração de envelhecimento de software obtido pela aplicação do método proposto neste trabalho.

1.4. DESENVOLVIMENTO DA PESQUISA

A fim de atingir os objetivos propostos na seção anterior, a seguir serão apresentadas as etapas para o desenvolvimento deste trabalho.

1.4.1. Revisão da literatura

Nesta fase se desenvolve um estudo de levantamento bibliográfico das seguintes áreas: falhas de software, envelhecimento de software e ensaios de vida acelerados. O primeiro tópico se faz necessário, a fim de se estudar em detalhes os mecanismos que causam as falhas de software. Um estudo aprofundado com vistas ao estado da arte é conduzido para o tópico de envelhecimento de software, visto ser este o objeto central da pesquisa. O terceiro assunto apresenta o arcabouço teórico e prático que foi adotado para reduzir o tempo de experimentação no que tange à obtenção dos tempos de falha por envelhecimento de software. O estudo deste tópico é fortemente baseado em literatura com aplicações em outras áreas do conhecimento, haja vista que foi encontrado apenas um trabalho aplicando a técnica de ensaios acelerados na área de software. Contudo, sempre que possível, far-se-á suas explanações à luz dos propósitos principais desta pesquisa, que objetiva a aplicação desta técnica para componentes de software.

1.4.2. Método para aceleração de falhas de envelhecimento de software

A partir dos fundamentos abordados na revisão da literatura, foi construído um processo sistêmico, baseado em experimentação, com o objetivo de acelerar a ocorrência de falhas causadas pelo envelhecimento de software. Desta forma, buscou-se reduzir o tempo necessário para a obtenção da distribuição de vida dos sistemas analisados. O método proposto tem como base à utilização das técnicas de ensaios de vida acelerados (ALT), ensaios de degradação acelerados (ADT) e projeto de experimentos (DOE – *Design of Experiments*), as quais serão contextualizadas a seguir.

As técnicas de ensaios acelerados (ALT/ADT) exigem a definição de um ou mais mecanismos pelo qual é possível acelerar a vida do sistema sendo investigado, ou seja, utilizando tais mecanismos é possível reduzir o tempo até a falha do sistema. Estes mecanismos são chamados, na literatura de ensaios acelerados, de estresse de aceleração. Neste trabalho, as falhas de interesse são aquelas provocadas pelo envelhecimento de

software, portanto, a definição do(s) mecanismo(s) usado(s) para acelerar o envelhecimento de software é parte essencial do método proposto. Neste estudo, convencionou-se adotar o termo fator de envelhecimento para representar os mecanismos que permitem acelerar o envelhecimento de software. Em termos práticos, o fator de envelhecimento representa um ou mais fatores que exercem a maior influência sobre o envelhecimento do sistema investigado. Portanto, pode-se dizer que o conceito de fator de envelhecimento está para o método proposto, assim como o conceito de estresse de aceleração está para a teoria de ensaios acelerados (ALT/ADT). O termo "fator" foi utilizado, haja vista que a seleção do fator de envelhecimento é baseada na técnica de DOE, a qual adota esta terminologia para identificar aquelas variáveis que influenciam um dado processo/sistema sob estudo.

No método proposto, a partir de um conjunto de fatores candidatos, normalmente baseados nos parâmetros da carga de trabalho do sistema, são avaliados quais destes fatores exercem maior influência sobre o envelhecimento de software. Esta abordagem se concentra na causa e não nos efeitos do envelhecimento, diferente do enfoque dado até o momento pelos demais trabalhos nesta área. Não se tem conhecimento de outras pesquisas que abordam o envelhecimento de software sob esta perspectiva, sendo este enfoque uma das principais contribuições desta pesquisa. Contemplando o cumprimento do primeiro objetivo específico, o método proposto utiliza a técnica de projeto de experimentos (DOE) objetivando oferecer um arcabouço sistemático para a identificação do fator de envelhecimento.

A partir da definição do estresse de aceleração do envelhecimento (fator de envelhecimento), o próximo passo é o planejamento e execução dos ensaios de aceleração do envelhecimento, denominados neste método de testes de envelhecimento acelerados (TEA). O planejamento do TEA considera diversos aspectos, tais como a seleção dos níveis de aceleração, regras de alocação das unidades de teste, estabelecimento de um modelo de relacionamento vida-estresse, dentre outros. Portanto, cada um destes elementos faz parte das definições do método e serão tratados em detalhes neste trabalho.

Com base nos resultados do TEA, a próxima etapa do método se refere à análise dos tempos de falha obtidos em níveis elevados de estresse. Esta análise objetiva estimar a distribuição de vida do sistema para o seu nível padrão de operação. O nível padrão de operação (nível de uso) é aquele projetado para a operação do sistema em regime normal de trabalho (sem aceleração). Como resultado desta análise, tem-se uma estimativa da distribuição dos tempos de falha do sistema sem os efeitos da aceleração do envelhecimento. Esta estimativa é obtida a partir de um modelo teórico, denominado de modelo de relacionamento vida-estresse, o qual é um dos requisitos para a análise de dados de ensaios

acelerados. Dentre os diversos modelos de relacionamento vida-estresse descritos na literatura, foi escolhido para aplicação pelo método proposto o relacionamento da potência inversa (IPL).

1.4.3. Estudo experimental

A última etapa deste trabalho é a realização de um estudo experimental com o propósito de avaliar o método proposto. Esta avaliação objetiva verificar a tese, defendida neste trabalho, de que o envelhecimento de software pode ser acelerado de forma planejada, permitindo que os dados resultantes desta aceleração possam ser analisados estatisticamente, por meio de um modelo que relacione os tempos de falha e os níveis de estresse definidos para o fator de envelhecimento.

Neste sentido, primeiramente será avaliada a aderência do modelo de relacionamento IPL aos dados resultantes da aceleração do envelhecimento de software (produto do TEA). Apesar do modelo IPL ser utilizado em diversas aplicações de ensaios acelerados, não foram encontrados trabalhos utilizando-o para ensaios acelerados com componentes de software, o que exige a sua avaliação experimental. Deste modo, a partir dos dados obtidos com a execução do TEA, será avaliada a aderência do IPL para ser usado na análise de dados de aceleração do envelhecimento de software, contemplando assim o segundo objetivo específico do trabalho.

Outro propósito deste estudo experimental é avaliar a acuracidade das estimativas obtidas a partir da distribuição de vida estimada pelo método proposto. Para tanto, as estimativas obtidas para o nível de uso serão confrontadas com observações de um experimento controlado em um nível representativo do nível de uso, com o objetivo de analisar quantitativamente a qualidade destas estimativas. Desta forma, pretende-se atender ao terceiro objetivo específico deste trabalho.

1.5. ESCOPO DA PESQUISA

Este trabalho sistematiza a utilização de um conjunto de técnicas, reconhecidas e consolidadas em diversas áreas do conhecimento, voltadas a estudos experimentais envolvendo a obtenção de dados de vida de sistemas. Nenhuma nova técnica está sendo proposta, mas sim, estão sendo adaptadas a uma nova área de aplicação, neste caso a engenharia de software, com ênfase para estudos experimentais em envelhecimento de

software. O trabalho se concentra nos subsídios teóricos e práticos para que estas adaptações sejam possíveis.

Concernente à utilização do DOE durante a etapa de seleção do fator de envelhecimento (ver seção 1.4.2), esta proposta assume que a seleção dos fatores e níveis do projeto experimental é de responsabilidade do experimentador. Em Jain (1991) e Montgomery (2005), fica evidente que a correta seleção destes elementos é fundamental para o sucesso do projeto experimental. No caso de experimentos de software, e não sendo diferente em outras áreas, o domínio do conhecimento da área em questão é uma característica inerente ao experimentador. Com relação à configuração do projeto experimental e análise dos seus resultados, ambos aspectos são contemplados pelo método proposto.

Em se tratando do planejamento dos ensaios acelerados para o envelhecimento de software, o presente estudo contribui introduzindo o conceito de fator de envelhecimento, o qual permite aplicar a teoria de ensaios acelerados também para estudos de envelhecimento/degradação de software. Para analisar os dados obtidos com a aceleração do envelhecimento, se faz necessário à adoção de um modelo de relacionamento vida-estresse, o qual é parte integrante da teoria de ensaios acelerados. O método não define um novo modelo para este propósito, mas sim adota um modelo já existente e aplicado para estudos em outras áreas. Como não se tem registro de aplicação do modelo escolhido para acelerar falhas de software, a verificação da sua adequação constitui uma parte importante deste trabalho.

Entende-se que a partir de um conjunto significativamente variado de aplicações deste método, em diferentes cenários e com diferentes tipos de software, se possa encontrar uma coleção de modelos de relacionamento que seja representativa para diversas aplicações no campo da engenharia de software. Neste sentido, a construção deste corpo de conhecimento é considerada uma evolução deste trabalho.

Durante a aplicação do método, são necessários conhecimentos básicos sobre técnicas estatísticas, principalmente voltadas para testes de aderência, métodos de estimação de parâmetros, análise de regressão linear simples, dentre outros correlatos. O método proposto assume que o experimentador possui familiaridade com estas técnicas, não abordando os detalhes dos procedimentos para suas implementações. Já com relação aos principais critérios (ex. tamanho da amostra) necessários para a aplicação destas técnicas, o método fornece os subsídios requeridos para defini-los.

Com relação aos resultados obtidos no estudo experimental, ressalta-se que os mesmos são específicos ao caso estudado, inclusive sendo sensíveis a aspectos ambientais do experimento (ex. versões de software, parâmetros de configuração, etc.). Contudo, o processo

estabelecido para obter estes resultados independe dos detalhes de implementação, o que permite generalizar a sua aplicação.

1.6. ESTRUTURA DO TRABALHO

O capítulo 2 apresenta uma revisão da literatura sobre falhas de software. Inicialmente, são discutidos conceitos básicos juntamente com a taxonomia adotada no trabalho. Posteriormente, a cadeia fundamental da *dependabilidade* é descrita, estabelecendo o relacionamento causal entre os conceitos de falta, erro e falha. Tais conceitos são essenciais para a compreensão das causas do fenômeno do envelhecimento de software, juntamente com os mecanismos de aceleração do envelhecimento que serão abordados neste trabalho. Uma revisão de estudos empíricos à cerca de análises de falhas de software também é apresentada, destacando diversos elementos que serão explorados nos capítulos subseqüentes.

No capítulo 3, o tema envelhecimento de software é apresentado de forma aprofundada. O principal objetivo do capítulo é descrever a "anatomia" deste fenômeno, juntamente com os avanços realizados neste campo. Inicialmente, tem-se uma introdução contextualizando a problemática do envelhecimento de software, definindo principalmente a natureza das falhas causadas por envelhecimento de software. Em seguida, uma revisão das principais pesquisas publicadas na área nos últimos dez anos é apresentada, com destaque para os trabalhos desenvolvidos recentemente. No decorrer deste capítulo, diversos exemplos práticos são apresentados a fim de ilustrar didaticamente os detalhes envolvidos na "física" das falhas por envelhecimento de software. Além disso, será apresentado um levantamento de casos reais envolvendo falhas por envelhecimento em produtos utilizados no mercado, a fim de ilustrar a abrangência e importância que esta problemática tem no contexto atual. Um levantamento de casos práticos, similar ao que será apresentado, não tem sido encontrado na literatura, o que se configura uma contribuição deste capítulo para o corpo de conhecimento neste campo de pesquisa.

O capítulo 4 apresenta os principais conceitos à cerca da técnica de ensaios acelerados, haja vista que esta técnica é uma parte essencial do método proposto. Inicialmente, destaca-se a aplicação dos ensaios acelerados nas diversas áreas da indústria, enfatizando as principais motivações e dificuldades em se adotar tal abordagem. Na seqüência do capítulo são apresentados os diversos tipos de ensaios acelerados, juntamente com os requisitos e limitações de cada um. Posteriormente, os dois enfoques de ensaios acelerados (ALT e ADT), necessários para a composição do método, são descritos em detalhes. Os principais trabalhos

referenciados neste capítulo são voltados para aplicações em outras áreas que não a engenharia de software. Após extensa busca na literatura, foi encontrado apenas um trabalho que aplica a técnica de ensaios acelerados voltada para software, o qual será apresentado em detalhes na revisão da literatura do capítulo.

O método proposto será apresentado no capítulo 5. Este foi estruturado na forma de quatro processos principais, e o capítulo 5 é basicamente a descrição de cada um destes processos em detalhes. Um complemento deste capítulo é o apêndice A, o qual apresenta um dos algoritmos definidos para o método. Alguns exemplos numéricos são utilizados no decorrer do capítulo, para ilustrar a aplicação de determinados procedimentos necessários aos processos sendo apresentados.

De forma a avaliar o método proposto, o capítulo 6 apresenta um estudo experimental voltado para a aceleração de falhas causadas pelo envelhecimento de software em um sistema servidor web amplamente utilizado atualmente. A comprovação da existência dos efeitos do envelhecimento sobre este software foi reportada em publicações anteriores, o que motivou a sua adoção neste trabalho.

As conclusões do trabalho são apresentadas no capítulo 7, juntamente com sugestões de trabalhos futuros complementares. A figura 1.2 apresenta uma visão geral da pesquisa.

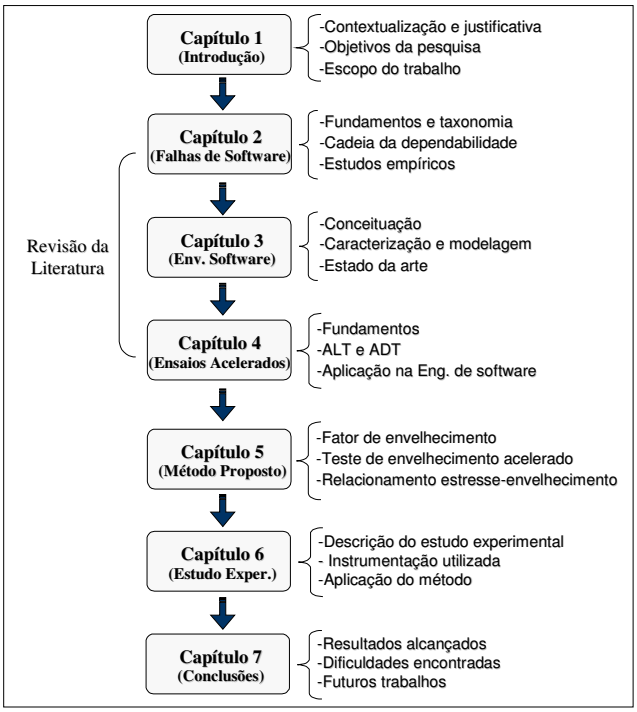

Figura 1.2: Estrutura do trabalho

CAPÍTULO 2

FALHAS DE SOFTWARE

2.1. INTRODUÇÃO

A falha de software é um dos principais elementos estudados na área de *dependabilidade* computacional. Os três elementos que compõem as ameaças à *dependabilidade* são: faltas, erros e falhas de software (figura 1.1). Este trabalho se concentra em uma classe de falhas em especial, denominadas falhas causadas pelo envelhecimento de software.

Devido aos diversos conceitos envolvidos, se faz necessário adotar uma terminologia bem estabelecida, a qual seja abrangente e atualizada, sendo compatível com a maior parte da literatura na área. Como já salientado na seção 1.1, são várias as terminologias propostas nesta área, principalmente com respeito às definições envolvendo os conceitos de falta, erro e falha. Buscando um comparativo destas definições com padrões em outras áreas da indústria, verificou-se que estes termos possuem diferentes semânticas dependendo dos autores e área de conhecimento. Como exemplo destas diferenças, as interpretações dadas aos termos *fault*, *error* e *failure*, por Jalote (1994, p. 6), são diferentes daquelas existentes na NBR 5462 (ABNT, 1994). Os quadros 2.1 e 2.2 apresentam as respectivas definições.

De acordo com a terminologia do quadro 2.1, obtém-se a seguinte relação causal: Falha $\Rightarrow$ Pane $\Rightarrow$ Erro, onde "$\Rightarrow$" indica o sentido da relação causa-efeito, assim como no exemplo: A causa B (A $\Rightarrow$ B). Com base nesta definição, percebe-se que o conceito de defeito poderia ser atribuído como a fonte causadora de falhas. Contudo, esta relação não está explicitamente definida na norma em questão.

Termo	Significado
Defeito	Qualquer desvio de uma característica de um item em relação aos seus requisitos. Um defeito pode ou não impedir um item de desempenhar sua função.
Falha	Término da capacidade do item desempenhar a função requerida. Depois da falha o item tem uma pane. A falha é um evento, diferente de pane que é considerado um estado.
Pane	Estado de um item caracterizado pela incapacidade de desempenhar uma função requerida. Uma pane é normalmente o resultado de uma falha.
Erro	Um erro pode ser causado por um item em pane. Por exemplo, um erro de cálculo feito por um computador em pane.

Quadro 2.1: Terminologia de confiabilidade da NBR 5462
Fonte: ABNT (1994)

Já em Jalote (1994), encontra-se explicitamente definida a seguinte relação de encadeamento: Falta (*fault*) ⇒ Erro (*error*) ⇒ Falha (*failure*), como pode ser observado no quadro 2.2.

Termo	Significado
Failure (falha)	Quando o comportamento do sistema desvia-se de sua especificação. A causa de uma falha é um erro.
Error (erro)	É a porção do estado do sistema responsável por conduzi-lo a uma falha.
Fault (falta)	A falta está associada à noção de defeito. A ativação de uma falta causa um erro.

Quadro 2.2: Terminologia de *dependabilidade* proposta por Jalote
Fonte: Jalote (1994)

Com relação ao termo "erro", em ambas terminologias seu significado é completamente distinto. Ressalta-se também, que a NBR 5462 é baseada no documento IEC 50-191[8], tendo como termos equivalentes "Pane" (NBR 5462) e "*Fault*" (IEC 50-191). Da mesma forma como no caso anterior, a definição de *Fault* da IEC 50-191 difere amplamente daquela descrita em Jalote (1994).

Sobre estas divergências entre terminologias, a comunidade científica na área de *dependabilidade* computacional tem realizado várias iniciativas no sentido de se reduzir tais inconsistências, a exemplo dos trabalhos de Veríssimo e Lemos (1989) e Avižienis, Laprie e Randell (2001). Recentemente, o trabalho intitulado "*Basic Concepts and Taxonomy of Dependable and Secure Computing*" (AVIŽIENIS *et al.*, 2004), o qual estende a publicação anterior do mesmo autor, tem sido recorrentemente citado como referência de taxonomia nesta área, o que levou a adoção desta referência no contexto deste trabalho. A seguir a descrição dos principais termos e conceitos da taxonomia adotada.

2.2. TAXONOMIA UTILIZADA

Em Avižienis *et al.* (2004), uma falha (*failure*) é um evento que ocorre quando a função realizada pelo sistema não está de acordo com a correta especificação que foi definida para a sua execução. Denomina-se a entrega correta do serviço quando a função foi realizada em conformidade com sua especificação. Uma falha, portanto, ocorre quando o serviço entregue desvia-se da sua correta especificação. Este desvio, por parte do serviço entregue, pode assumir diferentes formas que são denominadas modos de falha do serviço.

[8] *International Electrotechnical Vocabulary. Chapter 191: Dependability and quality of service* (IEC 50-191 apud ABNT, 1994).

Um serviço entregue por um sistema é o seu comportamento como este é percebido pelos seus usuários. O comportamento de um sistema é o conjunto completo dos seus estados (internos e externos). A porção do comportamento do sistema, percebida pelos seus usuários, é denominada de estado externo. Correlacionando estes conceitos, tem-se que um serviço é a seqüência de estados externos do sistema. A falha de um serviço significa que no mínimo um ou mais estados, da seqüência de estados externos, desviou-se da sua correta especificação. Este desvio é denominado um erro (*error*). A suposta causa de um erro é considerada uma falta (*fault*).

Faltas podem ser internas ou externas ao sistema. Avižienis *et al.* (2004) enfatizam que, em muitos casos, uma falta primeiramente provoca um erro no estado de um componente que é parte do estado interno de um sistema, não afetando imediatamente o seu estado externo. Portanto, esta falta não obrigatoriamente causa o desvio da função (serviço entregue). Deste modo, um erro é definido como a parte do estado total do sistema que pode, mas não necessariamente irá, levar a uma subseqüente falha do sistema. Muitos erros poderão não alcançar o estado externo do sistema a fim de causar uma falha.

Uma falta que provocou um erro denomina-se falta ativa. Faltas que não foram ativadas são chamadas de faltas dormentes ou latentes. Quando a especificação funcional do sistema inclui um conjunto de várias funções, a falha de um ou mais serviços que implementam estas funções pode levar o sistema a um modo de degradação, o qual entrega apenas parte dos seus serviços. Neste caso, se diz que o sistema sofre de uma falha parcial.

Um aspecto importante ao se estudar as falhas de software é a sua relação com o ciclo de vida do software. Basicamente, o ciclo de vida consiste de duas fases principais: desenvolvimento e operação (uso). Na primeira, o sistema interage com o ambiente de desenvolvimento (ex. programadores, ferramentas de desenvolvimento, mundo físico, outros), onde podem ser introduzidas faltas de desenvolvimento. A fase operacional da vida do sistema se inicia quando o software é aceito para uso, tendo início o seu funcionamento junto aos seus usuários. Nesta etapa o sistema interage com o ambiente de produção (de uso), o qual consiste de diversos elementos (ex. mundo físico, usuários, administradores do sistema, provedores de serviço/recursos para o software, dentre outros). Além das faltas de operação, características da fase operacional, têm-se também ações de manutenção no sistema, as quais envolvem modificações no software e, portanto, muitas das interações que ocorrem na etapa de desenvolvimento se repetem nesta fase.

A seguir será apresentada uma classificação à cerca dos conceitos de falta, erro e falha, haja vista suas importâncias para os capítulos subseqüentes.

2.2.1. Classificação de faltas

Segundo Avižienis *et al.* (2004), as faltas que podem afetar um sistema durante sua vida são classificadas de acordo com oito critérios, os quais estão apresentadas na figura 2.1.

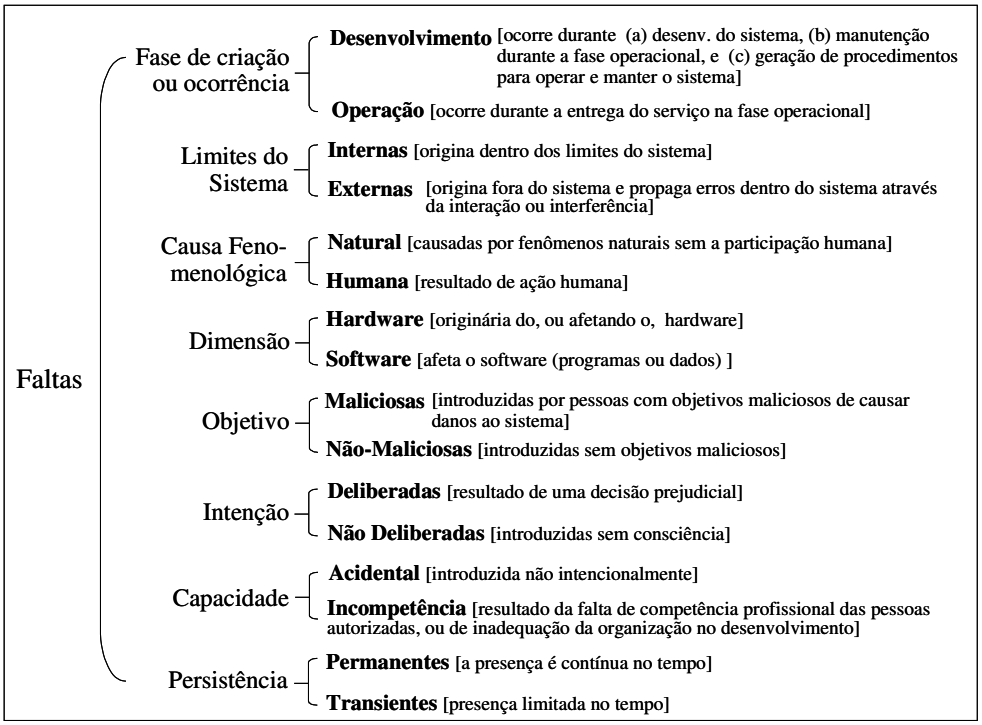

Figura 2.1: Critérios de classificação de faltas de software
Fonte: Traduzido de Avižienis *et al*. (2004)

Avižienis *et al.* (2004) explica que se todas as oito combinações de classes elementares fossem possíveis, existiriam 256 combinações, contudo nem todos os critérios são aplicáveis a todas as classes de falta. Neste caso, os autores sugerem 31 combinações possíveis, ressaltando que novas podem ser identificadas no futuro. As combinações são baseadas nas classes listadas na figura 2.2, juntamente com os três grupos de classes de faltas considerados representativos pelos autores, a saber: faltas de desenvolvimento, faltas físicas e faltas de interação. A figura 2.2 apresenta estas combinações na forma matricial.

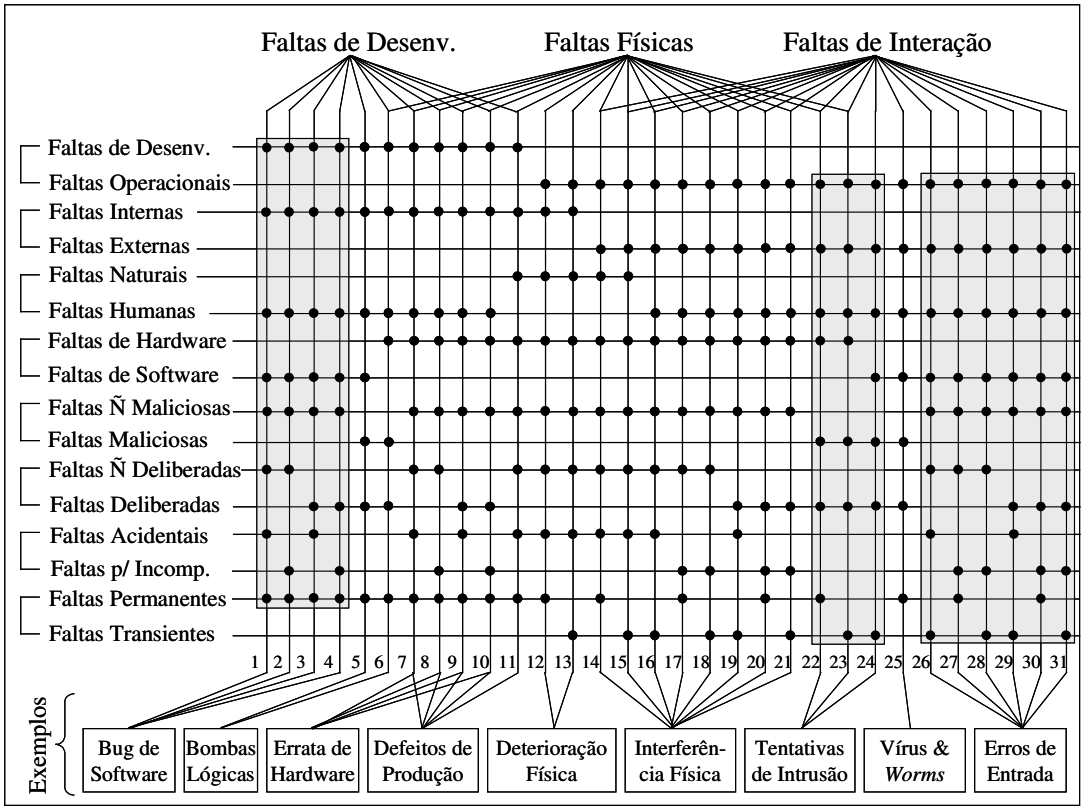

Figura 2.2: Representação matricial das combinações de classes de faltas
Fonte: Traduzido e adaptado de Avižienis *et al*. (2004)

A versão original da figura 2.2 não contém as regiões sombreadas como apresentado neste documento. Esta contribuição visa destacar aquelas classes de faltas que, direta ou indiretamente, possuem relação com o fenômeno do envelhecimento de software (ver capítulo 3). Verifica-se que as faltas relacionadas ao envelhecimento estão presentes, em maior intensidade, naquelas interações voltadas ao desenvolvimento. Outro aspecto importante é quanto à causa fenomenológica destas faltas, que são basicamente de origem[9] humana. A figura 2.3 apresenta a classificação das faltas de origem humana.

As classes de faltas de origem humana que exibem relação com as faltas relacionadas ao envelhecimento são as seguintes: #1-#4, #22 - #24 e #26 - #31. Na seção 3.2.1, serão apresentados exemplos que ilustram algumas destas classes contextualizadas em cenários reais.

[9] De acordo com a taxonomia adotada, a causa de uma falta é uma falha, neste caso, uma falha de origem humana.

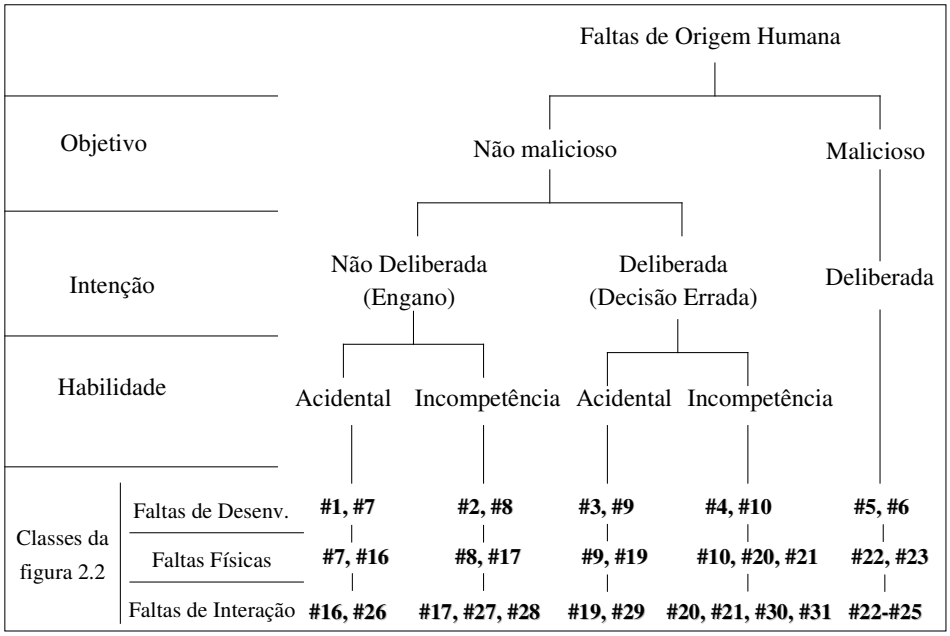

Figura 2.3: Classificação de faltas humanas
Fonte: Traduzido de Avižienis *et al.* (2004)

2.2.2. Classificação de falhas

Em Musa (1987, p. 8), o conceito de falha é definido como o desvio dos resultados externos da operação de um programa com relação aos seus requisitos. Esta definição, assim como aquela descrita em Jalote (1994, p. 6), corroboram com o conceito de falha adotado neste trabalho (ver seção 2.2).

De acordo com Avižienis *et al.* (2004), um aspecto importante relacionado às falhas de software são os modos de falha, os quais caracterizam os desvios que causam a incorreta entrega do serviço. Estes são definidos de acordo com 4 critérios:

- o domínio da falha;
- a "detectabilidade" da falha;
- a consistência da falha;
- as conseqüências da falha sobre o ambiente de operação do software.

Com relação ao domínio da falha, tem-se:

- falhas de conteúdo: o conteúdo da informação entregue na interface do serviço desvia-se do requerido na especificação;
- falhas de temporização: o tempo de chegada ou a duração da informação entregue na interface do serviço desvia-se do requerido.

Estas definições podem ser especializadas, como por exemplo, no caso das falhas de conteúdo serem do tipo numérico ou não numérico (ex. alfabético, gráfico, etc.). Quanto às falhas de temporização, podem ser tanto do tipo falhas por antecipação quanto de atraso, ambas com relação ao tempo especificado para a correta entrega do serviço.

Em ambos os casos as falhas situam-se em duas classes:

- falha de paralisação (*halt failure*): quando o serviço sendo prestado é paralisado, ou seja, o estado externo se torna constante não havendo mais atividade do sistema na interface com seus usuários. Um exemplo deste tipo de falha no cotidiano poderia ser a ausência de resposta de um software ("travamento");
- falha de inconsistência (*erratic failures*): diferentemente da anterior, o serviço é entregue, porém de forma inconsistente.

Com relação ao critério de "detectabilidade", este considera a sinalização da falha do serviço para o usuário. A sinalização é gerada a partir de mecanismos no próprio sistema que checam o serviço entregue contra a sua especificação. A geração de alarmes do tipo falso-positivo ou falso-negativo são por sua vez, modos de falha dos próprios mecanismos de segurança.

Quanto ao critério consistência, tem-se a seguinte divisão:

- falhas consistentes: a entrega do serviço incorreto é percebida da mesma forma por todos os usuários do sistema;
- falhas inconsistentes: alguns ou todos os usuários percebem, de forma diferente, o serviço incorreto. É possível que alguns usuários percebam o serviço como correto.

Alguns trabalhos (JALOTE, 1994; LAMPORT; SHOSTAK; PEASE, 1982 apud AVIŽIENIS *et al.,* 2004) têm usado a definição de falhas Bizantinas (*Byzantine failures*) para representar esta classe de falhas.

As conseqüências ocasionadas pela falha estão diretamente relacionadas à severidade de cada falha. Existem diversos aspectos que devem ser considerados ao se definir os níveis de severidade, sendo que os principais consideram a disponibilidade (tempo de inatividade do sistema), confidencialidade (o tipo de informação indevidamente descoberta), integridade (extensão da corrupção dos dados e a capacidade de recuperá-los), e proteção (possível risco a vidas humanas). A escala de severidade deve estar dentro dos limites dos dois níveis definidos a seguir:

- falhas negligenciáveis (*minor failures*): quando as conseqüências da falha possuem custos similares aos benefícios do provimento do serviço corretamente;

- falhas catastróficas: quando os custos das conseqüências da falha são muitas ordens de magnitude, ou mesmo incomensuravelmente, superiores aos benefícios em prover o serviço corretamente.

A figura 2.4 sumariza os modos de falha descritos anteriormente.

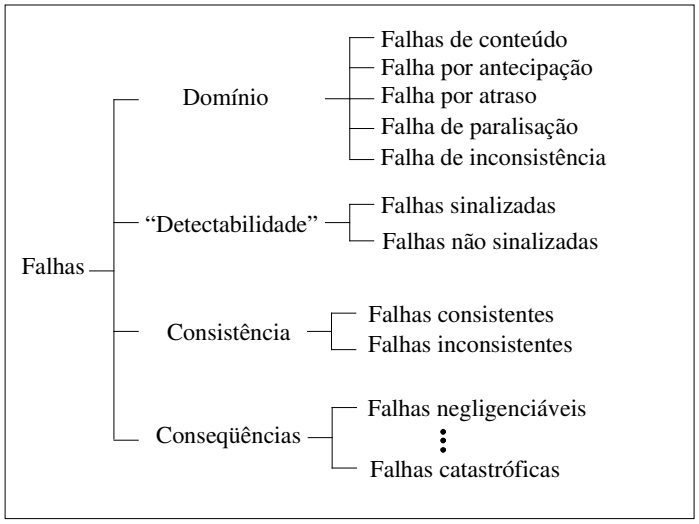

Figura 2.4: Modos de falha
Fonte: Traduzido de Avižienis *et al.* (2004)

2.2.3. Classificação de erros

Desde que o erro é uma propriedade do estado do sistema, este pode ser observado e avaliado. Falha, ao contrário, não é uma propriedade do estado do sistema, não podendo ser observada facilmente, salvo em alguns casos ao se utilizar instrumentação especial para registrar determinados tipos de eventos no sistema.

Tipicamente, a ocorrência de uma falha é deduzida detectando erros no estado externo do sistema (AVIŽIENIS *et al.*, 2004). A declaração de que uma falha ocorreu se dá verificando a ocorrência destes erros (JALOTE, 1994, p. 6). Quando o erro não afeta a correta entrega do serviço, ou seja, o estado externo do sistema, não se tem ocorrência da falha, apesar da existência do erro em alguns dos componentes internos ao sistema.

Quando um erro irá ou não causar uma falha no serviço prestado dependerá de dois fatores:

a) a estrutura do sistema, especialmente com relação a qualquer tipo de redundância existente, podendo ser:

- redundância intencional: implementada intencionalmente para fornecer tolerância à falta e evitar que um erro cause a falha do sistema;
- redundância não intencional: algum tipo de redundância presente no projeto do sistema, que apesar de sua existência não planejada, oferece um certo tipo de proteção ao sistema.

b) o comportamento do sistema: a porção do estado do sistema que contém o erro pode não ser necessária para um dado serviço, ou o erro pode ser eliminado antes de se propagar para o estado externo a fim de causar uma falha.

Com relação ao primeiro fator (a), um exemplo típico de redundância intencional são os aglomerados (*clusters*) de computadores para fins de alta disponibilidade. No caso da redundância não intencional, se podem citar as arquiteturas de software baseadas em múltiplos processos/*threads* denominadas de MPM (*Multiprocessing Modules*) (MENASCÉ, 2003; QIN; TUCEK; ZHOU, 2005), cuja própria organização dos seus componentes de software introduz naturalmente algum nível de redundância. Um exemplo de software que adota uma arquitetura baseada em MPM é o servidor web Apache[10].

No caso do segundo fator (b), tem-se como exemplo os mecanismos de rejuvenescimento (HUANG *et al*., 1996; VAIDYANATHAN; TRIVEDI, 2005), usados para mitigar os efeitos do envelhecimento de software.

Em Avižienis *et al.* (2004), recomenda-se classificar os erros de acordo com aquelas falhas (elementares) de serviço que eles causam, seguindo a mesma terminologia descrita na seção 2.2.2. Deste modo, tem-se erros de conteúdo, por atraso ou antecipação no tempo, erros detectados, não detectados (latentes), inconsistentes ou consistentes, negligenciáveis e catastróficos.

2.2.4. Cadeia fundamental da dependabilidade

Os três conceitos (falta, erro e falha) apresentados anteriormente especificam seus possíveis relacionamentos entre si. Para cada definição foi expressa uma relação de causalidade entre estes, que estão ilustradas na figura 2.5.

A ativação de uma falta pode ser entendida como a aplicação de um padrão de entrada (padrão de ativação) para um componente, fazendo com que uma falta dormente naquele

[10] Válido para versões 2.0 e superiores.

componente se torne ativa. A ativação da falta (interna e/ou externa) produz um erro que, ao se propagar, pode alcançar a interface externa do componente causando então uma falha no serviço prestado pelo componente.

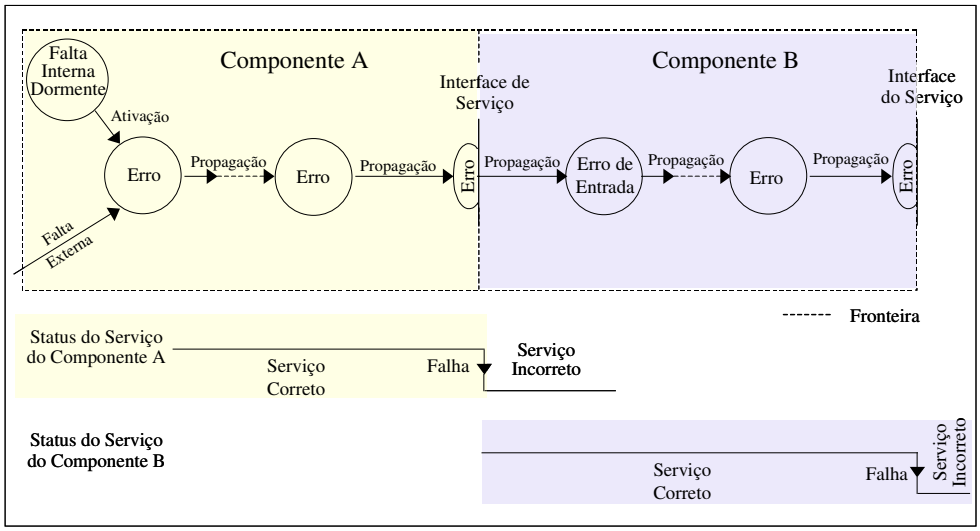

Figura 2.5: Propagação do erro
Fonte: Traduzido e adaptado de Avižienis *et al.* (2004)

Para o componente (B), usuário do serviço incorreto, a falha do componente provedor (A) aparece como uma falta externa ao componente (B). Caso um erro se propague até a fronteira do sistema, ou seja, sendo parte do estado externo do sistema, considera-se então uma falha do sistema. Da mesma forma, esta falha poderá ser uma falta externa para outros usuários na fronteira deste sistema. Como já descrito na seção 2.2.3, dependendo da estrutura e comportamento do sistema, o erro poderá ou não ficar confinado ao estado interno do componente e/ou sistema.

Para ilustrar este encadeamento, toma-se como exemplo o caso citado no capítulo 1, sobre o incidente envolvendo o equipamento médico Therac-25. O serviço prestado pelo equipamento é a sessão de terapia por radiação programada para cada paciente. Os usuários do serviço foram os diversos programas de tratamento de câncer, baseados em terapias radioativas, dos hospitais onde o equipamento se encontrava. A falha daquele dispositivo encadeou-se na forma de uma falta externa aos programas de tratamento de câncer. Tal falta introduziu erros nos tratamentos em andamento, os quais culminaram nas falhas dos mesmos, sendo que em três casos estas falhas foram catastróficas (perdas de vidas humanas).

De um ponto de vista macro (ex. na percepção dos familiares das vítimas), a falha ocorreu na prestação do serviço pelo hospital. Analisando a causa raiz do problema (visão

micro), pode-se dizer que um erro na programação do software do Therac-25 causou uma falha na codificação do programa, introduzindo uma falta dormente no código deste software. Esta falta foi ativada, a partir de um padrão de entrada específico durante a operação do equipamento pelo operador humano, resultando assim no encadeamento que levou à falha anteriormente descrita.

A partir deste exemplo, salienta-se que a cadeia de *dependabilidade* pode ser abordada em diversos níveis de visão, dependendo do ponto de vista que são considerados os conceitos de provedor e usuário do serviço. A figura 2.6 apresenta uma visão geral da cadeia fundamental da *dependabilidade* como proposto em Avižienis *et al.* (2004). As setas representam a relação de causalidade entre faltas, erros e falhas.

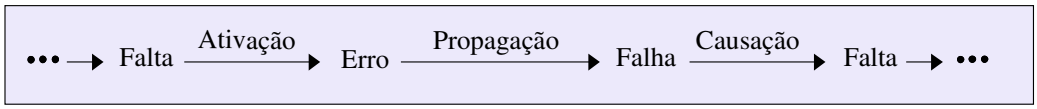

Figura 2.6: Cadeia fundamental da *dependabilidade*
Fonte: Avižienis *et al.* (2004)

A possibilidade de se identificar o padrão de ativação de uma falta é denominada reprodutibilidade da ativação de faltas. De acordo com esta propriedade, as faltas podem ser categorizadas em: sólidas (*solid*/*hard*) e obscuras (*elusive*/*soft*). As primeiras são aquelas faltas reproduzíveis sistematicamente. Já a segunda categoria não é sistematicamente reproduzível, normalmente tendo seu padrão de ativação dependendo de complexas combinações entre o estado interno do software e eventos externos, os quais ocorrem raramente e por isso sendo de difícil reprodução (GRAY, 1986).

A similaridade das manifestações das faltas de desenvolvimento caracterizadas como obscuras e das faltas transientes, levam ambas a serem agrupadas, compondo uma classe denominada de faltas intermitentes. No estudo das causas do envelhecimento de software, faltas intermitentes possuem uma grande importância e serão novamente abordadas no capítulo 3.

Ainda com relação às categorias de faltas do tipo sólidas e obscuras, uma terminologia anterior ao trabalho Avižienis *et al.* (2004) foi definida em Gray (1986). Em seu trabalho, Jim Gray utiliza os termos *Bohrbugs* e *Heisenbugs*, os quais são equivalentes aos termos faltas sólidas e obscuras/intermitentes, respectivamente. Esta terminologia de Gray tem sido preferencialmente utilizada pela literatura na área de envelhecimento de software. Deste modo, a fim de manter compatibilidade com a literatura da área, em especial com relação a

esta parte da taxonomia, os termos adotados neste trabalho serão *Bohrbugs e Heisenbugs* para representar faltas sólidas e obscuras/intermitentes, respectivamente.

A seguir será apresentada uma síntese de alguns trabalhos revisados que caracterizam a ocorrência de falhas de software, oferecendo evidências empíricas sobre diversos aspectos que serão associados ao problema do envelhecimento de software no próximo capítulo.

2.3. ESTUDOS EMPÍRICOS SOBRE FALHAS DE SOFTWARE

Os trabalhos que são discutidos nesta seção servem de indicativo da problemática das falhas de software, bem como oferecem algumas tendências de padrões de ocorrência que serão explorados no próximo capítulo.

Em Basili e Perricone (1984), uma investigação detalhada foi conduzida com os dados de alterações (manutenções) de código realizadas no âmbito de um projeto de software de médio porte, contendo aproximadamente 90.000 linhas de código fonte em linguagem Fortran para computadores IBM 360. O software analisado foi desenvolvido para estudos de planejamento de satélites. Os dados foram coletados durante um período de 33 meses. Como resultados obtidos, os autores verificaram que erros relacionados à especificação funcional, bem como de requisitos do software, corresponderam a 48% do total de erros encontrados. Dos demais erros catalogados, 24% foram relacionados ao projeto (2%) e programação (22%). Em especial sobre a maior parcela de erros relacionar-se com a fase de especificação, outros trabalhos na área (CHILLAREGE; KAO, CONDIT, 1991; VAN DER MEULEN; BISHOP; REVILLA, 2004) têm apresentado conformidade com estes resultados.

Basili e Perricone (1984) chamam a atenção para o fato de que, dos 48% dos erros relacionados à especificação e requisitos, 28% envolveram módulos já existentes que foram modificados e 20% envolveram novos módulos. Com isso, os autores concluíram que a reutilização de código não necessariamente reduz os custos de desenvolvimento, podendo ocasionar retrabalho devido aos erros introduzidos durante a reutilização. Verificou-se, também, que dentre diversas classes de erros, 39% relacionaram-se à interface do sistema, 19% à computação (ex. operações aritméticas e lógicas) e 16% ao controle de fluxos (ex. *if-then-else*). Além das análises citadas, os autores correlacionaram a ocorrência dos erros com o tamanho dos módulos e suas complexidades[11]. Ao contrário do senso comum, na medida que

[11] A definição de complexidade seguiu o conceito de *cyclomatic complexity* descrita em Schneidewind (1979 apud BASILI; PERRICONE, 1984).

o tamanho dos módulos aumentou a complexidade aumentou, porém o número de erros diminuiu.

A relação dos resultados citados anteriormente, com as falhas causadas por envelhecimento de software, está na maior parcela de faltas relacionadas à etapa de desenvolvimento. Como destacadas na figura 2.2, as faltas de envelhecimento estão muito relacionadas às falhas de origem humana.

A maioria dos estudos envolvendo faltas, erros e/ou falhas de software, tal como em Basili e Perricone (1984), têm se voltado para a fase de desenvolvimento, visando reduzir, tanto quanto possível, faltas que poderiam causar falhas na etapa operacional do software (SULLIVAN; CHILLAREGE, 1991). As falhas de software que ocorrem em um ambiente de produção, muitas vezes possuem características diferentes daquelas encontradas na fase de desenvolvimento (ex. teste do produto). Mudanças ambientais (ex. SO, hardware), variações na carga de trabalho, ou uma combinação destas entre si ou com outros possíveis fatores, podem ativar faltas latentes que dificilmente se manifestariam em testes de laboratório.

O trabalho de Sullivan e Chillarege (1991), engloba a análise de dados de três anos de relatórios de campo relativos à falhas de um software de sistema operacional da IBM. O estudo se concentrou na análise de basicamente dois tipos de erros, os quais os autores denominaram de erros regulares e de memória. São exemplos de erros regulares: operações aritméticas e lógicas, controle de fluxo, sincronização, dentre outros. Os erros de memória se referiram a qualquer erro relacionado à corrupção de memória. A decisão em se analisar estes tipos de erros foi tomada a partir de entrevistas com especialistas de atendimento em campo da IBM, os quais freqüentemente se deparavam com falhas de sistema provocadas por tais erros.

Sullivan e Chillarege analisaram tanto as faltas causadoras dos erros quanto os padrões de ativação destas faltas.Verificou-se que em todas as amostras de relatórios analisadas, os erros de memória possuíam em média três vezes mais ocorrências do que os demais tipos de erro. Aprofundando a análise dos erros de memória, visto sua maior "pervasividade", os autores buscaram caracterizar as faltas causadoras destes erros. Os três tipos de faltas de maior ocorrência tinham relação com: alocação de memória, estouro de buffer (*copying overrun*) e gerenciamento de ponteiros. Dois destes, o primeiro e o terceiro, têm forte relação com o problema de vazamento de memória (*memory leaks*), o qual tem sido freqüentemente citado em pesquisas voltadas ao envelhecimento de software. O vazamento de memória será descrito em detalhes no capítulo 3.

Ainda com relação aos erros de memória, verificou-se que os eventos de ativação das faltas que provocaram estes erros foram, na sua maioria, eventos de tratamento de exceções (*error handling*), condições-limite e interrupções de execução[12]. Uma importante consideração é que todos os três casos podem ser considerados eventos raros, ou seja, pouco freqüentes. Tal observação corrobora com os resultados apresentados em Hecht (1993), o qual realizou uma análise dos dados de falhas de software do projeto DSN (*Deep Space Network*), desenvolvido no laboratório JPL da NASA.

O software analisado em Hecht (1993), tinha como principal propósito estabelecer a comunicação e a recepção de dados de telemetria de espaçonaves. O estudo, restrito à fase operacional do software, verificou que durante o primeiro ano de operação ocorreram 33 falhas em códigos freqüentemente executados (CFE) e 42 falhas em códigos não freqüentemente executados (CNFE). Complementarmente, o autor analisou outro projeto de software da mesma instituição. Nesta segunda análise, foram estudados os dados de testes do software de controle de aviação da aeronave *Space Shuttle* da NASA. Os resultados, baseados nos relatórios de falhas, indicaram que 60% das falhas representavam falhas de CNFE. Juntamente com estes resultados, o trabalho referencia outras publicações que apresentaram resultados similares.

Hecht conjeturou que a inabilidade para tratar mais do que um evento raro no tempo (ex. uma exceção ocorrendo durante uma variação abrupta na carga de trabalho), durante a etapa de testes, seria uma das principais explicações para o padrão de comportamento encontrado nos dados que ele analisou. Esta dificuldade estaria presente em todas as etapas do desenvolvimento do software, fazendo com que as faltas ativadas por estas combinações de eventos permanecessem latentes no produto final, se manifestando na etapa operacional. As condições de operação, definidas por Hecht como eventos raros (CNFE), aparecem com freqüência na literatura de envelhecimento de software como será apresentado no capítulo 3. Outros dois estudos que também analisaram dados de falha na fase operacional do software são Trachtenberg (1992) e Fenton e Ohlsson (2000).

No primeiro (TRACHTENBERG, 1992), tem-se uma análise dos dados de falha de nove sistemas da IBM, publicados em Adams (1984 apud TRACHTENBERG, 1992). Adams verificou, experimentalmente, que a taxa de falha dos dados analisados seguia a Lei de

[12] As interrupções de execução são definidas pelos autores como um caso especial de condição limite, onde uma interrupção não desejada ocorre durante a execução do sistema operacional, no decorrer de uma seqüência de eventos específica, causando um erro no estado interno do sistema em questão.

Phillip. Segundo esta lei, o número de faltas de uma determinada taxa de falha é inversamente proporcional à potência da taxa de falha, como descreve a equação 2.1.

$$N = \frac{K}{\lambda^q}, \tag{2.1}$$

onde:

N = fração de faltas em um sistema com uma determinada taxa de falhas;

K, q = constantes peculiares a cada sistema;

λ = taxa de falha.

A fim de ilustrar a sua aplicação, a figura 2.7 apresenta o ajuste de um modelo de regressão de potência, por mínimos quadrados, utilizando os mesmos dados do quadro 1 que consta em Trachtenberg (1992).

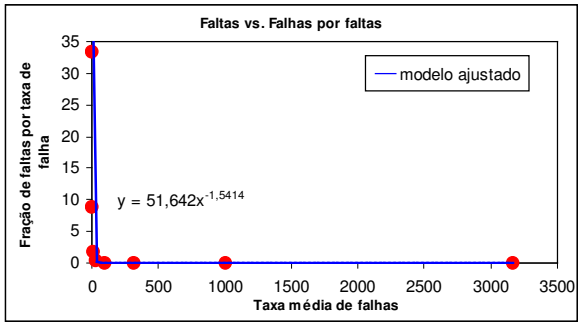

Figura 2.7: Modelo teórico de Phillip para os dados de faltas e falhas

O modelo ajustado na figura 2.7 segue a Lei de Phillip, onde y e x correspondem à N e λ respectivamente. No modelo ajustado, a taxa de falha ao ser levada para o denominador altera o sinal do seu expoente, resultando na expressão 2.1.

Trachtenberg (1992) demonstra matematicamente que a Lei de Phillip é uma conseqüência da Lei de Zipf[13] (LI, 2005). A Lei de Zipf diz que a freqüência de ocorrência de um evento E (neste caso uma falha), como função do seu ranking (i) (o ranking é determinado pela referida freqüência de ocorrência), é uma função da lei de potência:

$$E(i) = \frac{1}{i^p}, \tag{2.2}$$

[13] Apesar de ter sido criada na área de lingüística, a Lei de Zipf tem sido usada (ADAMIC; HUBERMAN, 2002) para explicar comportamentos existentes na Internet, especialmente com relação a questões envolvendo a web. Em Trachtenberg (1992) esta lei é proposta para modelar o comportamento da relação falta e falha na fase operacional do software.

com o expoente *p* tendendo a 1.

As conclusões de Adams e Trachtenberg são complementares. Adams verificou que uma grande proporção das faltas latentes causa falhas raras, assim como a grande maioria das falhas é causada por uma pequena proporção de faltas. Deste modo, remover uma grande quantidade de faltas pode ser pouco efetivo, pois somente quando a pequena proporção de faltas que causa a maior quantidade de falhas é removida se tem uma melhora significativa na confiabilidade do software.

Trachtenberg corrobora com Adams, demonstrando formalmente que a busca pela otimização dos esforços de desenvolvimento causa um comportamento observado pela Lei de Zipf, já que se verifica um padrão compatível com o princípio do mínimo esforço nos dados analisados. Estas evidências justificam os esforços atuais na área do envelhecimento de software, haja vista que eventos raros são de difícil detecção e que mesmo pequenas quantidades de faltas ativadas por tais eventos podem estar relacionadas a vários modos de falha.

No segundo trabalho (FELTON; OHLSSON, 2000), tem-se um estudo sobre os dados de falha de um sistema de software industrial. Os dados usados no estudo são provenientes de duas versões consecutivas de um grande sistema voltado para centrais telefônicas. Este desenvolvimento envolveu mais de 20 centros de projeto espalhados em mais de 10 países. Cada versão demorou em média dois anos para ser desenvolvida, consumindo mais de duzentos homens-ano. O número de módulos selecionados aleatoriamente para a análise, foi de 140 e 246 para a primeira e a segunda versão do software, respectivamente. O tamanho dos módulos variou de 1.000 a 6.000 linhas de código. Foram testadas diversas hipóteses a partir da análise quantitativa dos dados. Uma síntese dos resultados destes testes está apresentada no quadro 2.3.

Os testes de hipótese foram realizados utilizando a técnica de diagramas de Alberg (OHLSSON; ALBERG, 1996 apud FELTON; OHLSSON, 2000). Os autores destacam a rejeição da hipótese 4 como um dos principais resultados do trabalho. Segundo Felton e Ohlsson, este resultado contraria algumas das atuais suposições teóricas, e crenças de senso comum, de que na busca pelos módulos que apresentam mais faltas na etapa operacional, são considerados como candidatos principais para serem revisados aqueles módulos que manifestaram o maior número de faltas na etapa de testes. Para os autores, as evidências de seu estudo demonstraram que nem sempre isso é válido, ou seja, a revisão também pode se iniciar por aqueles módulos com menores números de faltas encontradas na fase de testes, como ocorreu neste caso.

N°	Hipótese	Obteve-se evidência no teste de hipótese?
1(a)	Um pequeno número de módulos contém a maioria das faltas descobertas durante a fase de testes.	Sim, evidência de 20-60 (20% dos módulos contêm 60% das faltas).
1(b)	Se a hipótese 1(a) é verdadeira isso implica que aqueles módulos constituem a maior parte do código do programa.	Não.
2(a)	Um pequeno número de módulos contém a maioria das faltas descobertas durante a fase operacional.	Sim, evidência de 20-80 (20% dos módulos contêm 80% das faltas relacionadas a falhas de operação).
2(b)	Se a hipótese 2(a) é verdadeira isso implica que aqueles módulos constituem a maior parte do código do programa.	Não, forte evidência da hipótese oposta.
3	A alta incidência de testes funcionais implica em alta incidência nos testes de sistema (integração).	Não, pouca evidência.
4	A alta incidência de faltas na etapa de testes implica em alta incidência de faltas na etapa operacional.	Não, fortemente rejeitada.
5(a)	Pequenos módulos são menos propensos a falhas que grandes módulos.	Não.
5(b)	Métricas de tamanho, tal como LOC (*Lines of Code*), são bons preditores do número de faltas na etapa de desenvolvimento de um módulo.	Não, pouca evidência.
5(c)	Métricas de tamanho, tal como LOC, são bons preditores do número de faltas na etapa operacional de um módulo.	Não.
5(d)	Métricas de tamanho, tal como LOC, são bons preditores da densidade de faltas[14] na etapa de desenvolvimento de um módulo.	Não.
5(e)	Métricas de tamanho, tal como LOC, são bons preditores da densidade de faltas na etapa operacional de um módulo.	Não.
6	Métricas de complexidade são melhores preditores do que métricas de tamanho.	Não, poucas evidências.
7	A densidade de faltas correspondente às fases de teste e operação é aproximadamente constante entre versões subseqüentes de um software.	Sim.
8	Sistemas de software produzidos em ambientes similares possuem geralmente similares densidades de faltas nas fases de teste e operacional.	Sim.

Quadro 2.3: Síntese dos resultados da pesquisa
Fonte: Felton e Ohlsson (2000)

Ao comparar estes resultados com aqueles apresentados em Trachtenberg (1992), verifica-se a existência de um padrão similar com respeito à relação falta-falha, sendo um forte indício da presença do princípio de Pareto[15] nos dados do trabalho. Este princípio possui estreita relação com as leis de Phillip e Zipf descritas anteriormente. O princípio de Pareto tem sido usado, intensivamente na engenharia de software experimental, para representar a distribuição de faltas por módulos, aplicando-se a chamada regra 80-20, ou seja, 80% das faltas estão contidas em 20% dos módulos, aproximadamente. Esta regra tem sido observada

[14] O número de faltas encontradas em um módulo dividido pelo tamanho do módulo em LOC.

[15] Inicialmente criada para explicar fenômenos da área sócio-econômica, a lei de Pareto em sua forma geral diz que 80% dos objetivos são conseguidos com 20% dos meios/recursos. No contexto da engenharia de software, e especificamente da taxonomia adotada neste trabalho, uma interpretação equivalente seria que os objetivos são os efeitos (falhas) e os meios de atingi-los são as causas (faltas).

também com relação aos aspectos dinâmicos do software, tais como nos trabalhos de Jalote (2004) e Ganapathi *et al*. (2004), assim como em Hecht (1993).

Jalote (2004) analisou os dados de falhas em campo (fase operacional) de diversos produtos de software da empresa Microsoft. Os dados foram obtidos de um sistema de monitoração remota de falhas, onde se verificou que, em muitos casos, a regra 80-20 ajustou-se também à distribuição falha-usuário, onde a maior porção das falhas ocorreram com uma pequena parcela dos usuários monitorados. Um resultado compatível com este é apresentado em Ganapathi *et al*. (2004), o qual considerou apenas falhas envolvendo o sistema de *Registry*[16] do sistema operacional Windows. Foram estudados 200 casos reais de falhas deste sistema, que ocorreram entre 1997 e 2003, verificando que a maior parte dos problemas tiveram pouca freqüência, enquanto uma pequena parcela ocorreu com um número significante de usuários.

Outro recente estudo envolvendo falhas de campo, específicas de ambiente *desktop*, é descrito em Ganapathi e Patterson (2005). O estudo coletou dados de aproximadamente 200 computadores pessoais (PC), pertencentes a departamentos da universidade de Berkeley. Todos os computadores executavam o sistema operacional MS Windows XP SP1. O trabalho se baseou em uma amostra de 1.546 registros de problemas. Destes, apenas 72 (4,65%) foram causados pelo sistema operacional, sendo os demais 1.474 (95,34%) tendo como causa raiz os processos de aplicação. A classe de aplicação que teve maior participação nos 95,34% de falhas foi a de navegação web (38%).

Em Chou *et al*. (2001), dois sistemas operacionais de código aberto e de livre distribuição (FOSS)[17] foram analisados, a saber: Linux (2.4.1) e OpenBSD (2.8). Como resultados, Chou *et al*. verificaram que os controladores de dispositivos (*device drivers*) possuíam a maior parte das faltas encontradas, em torno de 70% com relação a outras partes do *kernel* (ex. gerenciamento de processador, sistema de arquivos, etc.). De forma geral, a taxa de erros presente nos *device drivers* ficou em torno de sete vezes mais que nos demais subsistemas. Verificou-se que, em média, uma falta permanece no código em torno de 1,8 ano. Com relação à distribuição das faltas por arquivos-fonte, novamente tem-se a presença do princípio de Pareto. Dos erros analisados, 100% destes estão presentes em apenas 11% dos arquivos-fonte. Complementarmente, verificando aquelas faltas que mais causam falhas

[16] A *Registry* é um repositório hierárquico onde são armazenados dados de configuração do sistema operacional MS Windows, bem como dos diversos programas instalados (GANAPATHI *et al*, 2004).

[17] Free/Open Source Software é um software cujo código fonte é disponível de forma livre através de uma licença de software aceita pela OSI (*Open Source Initiative*) e/ou FSF (*Free Software Foundation*).

durante a execução do sistema operacional, apenas 10% dos arquivos são responsáveis por estas faltas.

2.4. CONSIDERAÇÕES FINAIS

Este capítulo introduziu a taxonomia que será utilizada nos próximos capítulos, com relação aos principais termos e conceitos envolvendo falhas de software. Foram também apresentados os fundamentos à cerca dos relacionamentos de cada elemento envolvido na cadeia fundamental da *dependabilidade*. As relações de causa e efeito foram discutidas, detalhando seus papéis como ameaças à *dependabilidade* computacional.

Complementarmente, foram apresentados estudos empíricos relacionados a falhas de software, abordando tanto os aspectos estáticos do software, através da análise de faltas em nível de código-fonte, quanto os aspectos dinâmicos, envolvendo falhas na fase operacional. A relativa baixa quantidade de pesquisas empíricas em engenharia de software, especificamente com relação à falhas de software, dificulta comparar estes resultados em diversos cenários, o que impede a generalização da maioria destes resultados. Contudo, alguns padrões de comportamento com relação às falhas de software têm sido verificados repetidamente nos trabalhos publicados. Por exemplo, têm sido encontradas evidências do princípio de Pareto nas relações falta-falha, falha-módulo e falha-usuário.

O próximo capítulo abordará em detalhes as falhas de software causadas por envelhecimento, as quais são uns dos principais elementos de investigação desta pesquisa. Diversos resultados apresentados neste capítulo possuem relação com as falhas de envelhecimento, principalmente aqueles envolvendo erros de memória. Em Sullivan e Chillarege (1991), tais erros foram predominantes nos sistemas estudados, chegando a apresentar três vezes mais ocorrências que os demais tipos de erro. Destacam-se também os resultados obtidos por estes mesmos autores, em conformidade com Hecht (1993), com relação ao padrão de ativação das faltas que ocasionaram os erros de memória. Verificou-se que a maioria destas faltas é ativada por uma pequena parcela de eventos, denominada por Hecht de eventos raros. A descrição destes eventos, em ambos trabalhos, está relacionada com os padrões de ativação das faltas relacionadas ao envelhecimento de software, que serão chamadas no capítulo 3 de *Heisenbugs*, conceito equivalente ao termo faltas intermitentes definido neste capítulo.

A revisão da literatura envolvendo falhas na etapa operacional considerou diversos tipos de software, desde sistemas de centrais telefônicas, controle de espaçonaves, sistemas

operacionais, editores de texto, navegadores de Internet, dentre outros. Em todos os casos, ficou evidente que apesar dos contínuos esforços em se ampliar a qualidade do software durante a fase de desenvolvimento, faltas remanescentes no software que entra em produção é um grande desafio para a *dependabilidade* computacional.

Neste sentido, algumas pesquisas (BROWN; PATTERSON, 2001; CANDEA, 2003) têm assumido que a falha de software é um fato inevitável e que a melhor abordagem é propor soluções para reduzir seu impacto sobre a *dependabilidade*, mitigando seus efeitos durante a fase operacional do software. Esta premissa parece se confirmar ao se considerar os diversos cenários do mundo real, em que mesmo comprovando a existência de uma falha em um produto de software, seus usuários muitas vezes precisam conviver com o problema aguardando que uma solução seja providenciada pelo fornecedor do produto. Em muitos casos, após reconhecer a existência do problema, o fornecedor do produto sofre influências de diversas ordens (ex. questões de ordens técnica, administrativa econômica, outras) que podem ou não indicar a viabilidade em se efetuar a correção do produto. Mesmo em situações de grande criticidade, muitas vezes a solução viável é conviver de forma apropriada com o problema, buscando minimizar seus efeitos quando este se manifesta.

Um exemplo que ilustra a abordagem dos autores supracitados, envolvendo o envelhecimento de software, foi o caso do sistema Patriot citado no capítulo 1. Após a detecção do problema, a solução definitiva não foi implementada em curto prazo, exigindo a convivência com os efeitos do envelhecimento, adotando uma abordagem paliativa durante a guerra do Golfo. Esta solução temporária foi a reinicialização do programa do sistema *Patriot*, em intervalos de tempo programados. Como mencionado anteriormente, esta solução mitigou os efeitos do envelhecimento evitando que o sistema falhasse na detecção dos mísseis Sculd, contudo, introduziu um "custo" que foi o período de vulnerabilidade referente à janela de tempo exigida para a inicialização ("*boot*") do sistema.

CAPÍTULO 3

ENVELHECIMENTO DE SOFTWARE

3.1. INTRODUÇÃO

O termo envelhecimento de software (*software aging*), atualmente tem sido usado para identificar duas linhas de pesquisa na área de engenharia de software. A diferença entre estas abordagens está no estado do software (estático ou dinâmico) em que ambas se aplicam.

Parnas (1994), quem primeiro adotou esta terminologia, trata o envelhecimento de software com relação aos aspectos estáticos do software, especialmente em nível de código fonte. Para Parnas, o envelhecimento de software é considerado sob a perspectiva do envelhecimento estrutural do programa, principalmente causado pelas mudanças ambientais (ex. plataforma de SO), bem como alterações em seu código através de manutenções para correções de problemas (*bugs*), intervenções evolutivas, dentre outras.

Durante o projeto do software não é possível antecipar todas as possíveis mudanças que ocorrerão ao longo do seu ciclo de vida, de forma que mudanças não previstas algumas vezes violam os princípios do projeto, resultando em uma manutenção de maior custo e muitas vezes com resultados insatisfatórios (YURCIK; DOSS, 2001, p. 49). Tal envelhecimento não ocorreria, se o ambiente operacional (ex. SO e hardware) para o qual o software foi originalmente construído não mudasse, da mesma forma como se o código do software não sofresse intervenções após sua entrada em produção. Porém, ambos cenários não são de forma alguma realistas.

Uma segunda linha de pesquisa tem focado em outra problemática que, apesar de ser diferente da contextualização anterior, também foi denominado de envelhecimento de software. O envelhecimento de software abordado neste trabalho segue esta segunda linha de pesquisa, a qual será descrita a seguir.

Neste segundo enfoque, o envelhecimento de software representa os efeitos degenerativos que ocorrem durante o estado dinâmico do software, ou seja, enquanto o software se encontra em execução na memória do computador. Em Huang *et al.* (1995), o termo envelhecimento do processo de software (*software process aging*) foi utilizado, pela primeira vez, para caracterizar este problema que ocorre em muitos programas de computador que executam por longos períodos de tempo de forma ininterrupta. Em se tratando do

software em seu estado operacional, ou seja, durante sua execução, Huang *et al.* utilizam o termo processo[18], tal qual é definido no contexto de sistemas operacionais (STALLINGS, 2005, p. 110).

Apesar do termo envelhecimento do processo de software representar, de forma mais fidedigna, esta segunda problemática, os diversos trabalhos subseqüentes à Huang *et al.* (1995) têm adotado simplesmente "envelhecimento de software". Devido à ampla utilização desta definição na literatura, esta terminologia já se encontra praticamente consolidada. Portanto, a distinção ao se referenciar ambas linhas de pesquisa, através da mesma terminologia, deve ser feita pela contextualização do problema, haja vista a ambigüidade do termo. Neste documento, deste ponto em diante, o termo "envelhecimento de software" será usado para tratar a segunda linha de pesquisa, tal como definida em Huang *et al.* (1995).

O fenômeno do envelhecimento de software deve ser entendido como sendo a degradação progressiva do estado interno de um processo, e/ou do seu ambiente operacional, durante sua execução, tendo como conseqüência a degradação da sua performance, e também podendo causar a sua falha (HUANG *et al.*, 1995; GARG *et al.*, 1998a). O envelhecimento de software é um problema real com conseqüências reais. A mais grave que se tem registro foi o caso Patriot (ver seção 1.2) que resultou na perda de vidas humanas.

Na indústria de TI, existem diversos casos onde este problema tem-se manifestado. Como exemplo, cita-se o problema reportado (CISCO SYSTEMS, 2000) pela empresa Cisco Systems, o qual afetou diversos dos seus produtos da linha Catalyst (2900, 4000, 5000, 6000, outros). O problema foi o envelhecimento do processo `telnetd`, o qual implementa o serviço TELNET nos produtos afetados (neste caso *switches* de rede). Como conseqüência em longo prazo, ocorria a exaustão da memória principal dos equipamentos afetados, causando a interrupção completa do seu funcionamento. A única solução, enquanto o problema não fosse corrigido pelo fornecedor, era a reinicialização (*reboot*) completa dos equipamentos afetados. A causa raiz foi detectada como sendo um problema de vazamento de memória (*memory leak*), o qual se manifestava a cada tentativa de conexão TELNET ao *switch* e cuja autenticação falhasse. Também, se verificou o mesmo problema naqueles casos onde a autenticação TELNET obtinha sucesso, mas a sessão possuía uma curtíssima duração (poucos segundos). Problemas como estes são de difícil detecção, pois ocorrem de forma aleatória e muitas vezes sendo notados somente após longos períodos de execução ininterrupta. Neste caso, por exemplo, a utilização de conexões TELNET se dá apenas para tarefas

[18] Os termos processo, processo de aplicação e processo de software serão usados com o mesmo significado neste contexto.

administrativas no equipamento, o que torna suas execuções pouco freqüentes dificultando seu diagnóstico. Situações como estas podem ser comparadas aos eventos raros (CFNE) descritos no capítulo 2.

Vale ressaltar, que o caso citado anteriormente tratou de um software embarcado. Nestes casos, a disponibilidade de recursos (ex. espaço em disco e memória) é restrita devido à natureza destes sistemas, exigindo do software o uso mais eficiente dos recursos disponíveis. Problemas de envelhecimento de software em sistemas embarcados tendem a se manifestar em um período menor do que em servidores e/ou demais sistemas convencionais, haja vista a maior disponibilidade de recursos nestes últimos.

Existem vários casos documentados de problemas causados por envelhecimento de software, como por exemplo, no sistema de bilhetagem da Lucent (HUANG *et al.*, 1995), no software de central telefônica da AT&T (AVRITZER; WEYUKER, 1997), no servidor web Apache (LI; VAIDYANATHAN; TRIVEDI, 2002), no sistema concentrador de *cable modems* da Motorola (LIU *et al.*, 2002), na plataforma X2000 para missões espaciais da NASA (TAI; ALKALAI, 1998), em estações de trabalho UNIX (VAIDYANATHAN; TRIVEDI, 1998), dentre outros.

O avanço das pesquisas na área de envelhecimento de software fica evidenciado ao se analisar a evolução das publicações nos últimos dez anos (ver seção 3.3). Segundo Trivedi (2004) “atualmente o principal esforço de pesquisa nesta área tem sido em fornecer uma base científica para o fenômeno do envelhecimento de software”. A próxima seção apresentará em detalhes os principais aspectos relacionados à caracterização do fenômeno do envelhecimento de software.

3.2. FALHAS POR ENVELHECIMENTO DE SOFTWARE

O envelhecimento de software pode ser caracterizado como o acúmulo de erros, durante a execução de um processo de software, causando a degradação progressiva do estado interno deste processo. De acordo com Vaidyanathan e Trivedi (2005), os elementos que compõem o estado interno de um processo de software são:

- estado volátil: pilha do processo e segmentos de dados (estático e dinâmico);
- estado persistente: arquivos utilizados pelo processo;
- ambiente do sistema operacional: todos os recursos usados pelo processo (ex. memória virtual, portas de comunicação, estruturas no *kernel*, outros).

O efeito acumulativo de determinadas faltas de software, quando ativadas e incidindo sobre qualquer um destes três elementos, no decorrer do tempo de vida do processo, tem como conseqüências à diminuição progressiva do desempenho do processo ou até mesmo sua falha. Isto explica o motivo da maior incidência do envelhecimento de software naqueles programas que executam ininterruptamente por longos períodos de tempo, tais como sistemas servidores (ex. banco de dados), embarcados (ex. *switches* de rede), dentre outros.

Nas demais áreas da indústria, os modos de falha relacionados com o desgaste ou envelhecimento são comuns, como por exemplo, na engenharia mecânica o desgaste de uma peça (ex. eixo). A diferença principal dos modos de falha nestas áreas, com aqueles presentes no envelhecimento de software, reside basicamente nos mecanismos que causam o modo de falha. A seguir serão abordados em detalhes as causas e efeitos das falhas por envelhecimento de software.

3.2.1. Causa e efeito

Haja vista que falhas por envelhecimento de software têm como característica principal serem precedidas de um processo degenerativo do estado interno do software, a quase totalidade dos trabalhos nesta área tem indicado, como causas principais desta degradação, a exaustão de recursos do sistema operacional, corrupção de dados e acúmulo de erros numéricos. Sendo que a primeira tem sido reportada com maior freqüência pelos trabalhos na área. Neste sentido, é importante analisar as faltas relacionadas ao envelhecimento cuja ativação resultam nos efeitos de degradação supracitados.

Em diversos trabalhos, a exemplo de Gross, Bhardwaj e Bickford (2002), Vaidyanathan e Trivedi (2005), Xie, Hong e Trivedi (2004) e Garg *et al.* (1996b), alguns erros têm sido citados de forma recorrente como sendo as causas primárias desta degradação, os quais serão descritos a seguir:

- vazamento de memória (*memory leaking/bloating*): provoca um incremento no uso da memória pelo processo. A freqüência de ocorrência destes erros, durante o período de execução do processo, resulta no aumento do tamanho do processo, causando a exaustão dos recursos de memória (memória RAM e espaço de paginação/*swap*) do sistema operacional, o que causa a degradação de desempenho e até mesmo a parada do sistema como um todo.

- conexões pendentes (rede e banco de dados): devido ao *timeout* de desconexão, que a maioria dos protocolos implementam, a ocorrência de interrupções abruptas em conexões estabelecidas, podem acarretar o acúmulo de conexões aguardando *timeout* para seu fechamento. Desta forma, o incremento destas conexões em estado de espera, causam a exaustão de estruturas internas do sistema operacional (ex. *sockets*) necessárias ao estabelecimento de novas conexões. Além disso, erros desta natureza podem, em muitos casos, terem como efeito colateral o vazamento de memória tanto nos processos clientes quanto nos servidores.
- arredondamento/precisão numérica (*round-off errors*): a movimentação de dados entre variáveis de tipos diferentes, cujo sentido da transferência pode ocasionar perdas de precisão (ex. de real para inteiro, de *unsigned* para *signed*), culmina em resultados incorretos. Além disso, existem situações onde o incremento nos valores de uma variável extrapola a sua capacidade de armazenamento, o que provoca a reinicialização da variável para o seu menor valor suportado.
- controle de concorrência (*unreleased file-locks*): a não liberação de estruturas de controle de concorrência (ex. semáforos), especialmente vinculadas à *threads* e arquivos, cuja freqüência de utilização é alta (ex. SGBD), pode esgotar os recursos do sistema operacional. Para estes casos (arquivos/*threads*), tais erros também podem provocar um aumento considerável na carga de trabalho do sistema, já que o número de estruturas de controle envolvidas em cada evento cresce progressivamente. Além da degradação da performance do sistema, e conseqüentemente do processo envolvido, o acúmulo destes erros causa a exaustão de recursos que podem ser necessários por outros processos.
- dados corrompidos (*data corruption*): possuem como escopo tanto a memória interna do processo (ex. pilha, segmento de dados), quanto arquivos (ex. índices e tabelas de banco de dados), bem como estruturas de "metadados" do próprio SO (ex. tabela de alocação de arquivos).
- fragmentação do sistema de arquivos: resulta na degradação da performance de operações de entrada/saída (E/S) em arquivos.

Cada situação descrita anteriormente pode ser conseqüência de uma ou mais faltas ativadas durante a execução do processo, observando que os efeitos de suas ocorrências em longo prazo são as causas das falhas por envelhecimento. A natureza degenerativa destas faltas, com respeito ao estado interno do processo, as qualificam como sendo faltas

relacionadas ao envelhecimento (*aging-related faults*) ou simplesmente FRE (VAIDYANATHAN; TRIVEDI, 2001a).

De acordo com a taxonomia descrita no capítulo 2, o quadro 3.1 apresenta alguns exemplos de possíveis encadeamentos que podem ocorrer e que estão relacionados ao envelhecimento de software. Na primeira coluna se tem o estágio em que a falha que originou a falta relacionada ao envelhecimento ocorre. Os três principais momentos citados foram nas etapas de projeto, codificação e operação. Em cada uma destas etapas, a ocorrência de uma falha pode introduzir faltas relacionadas ao envelhecimento. Estas faltas (segunda coluna), quando ativadas, levam o sistema a cometer erros de execução (terceira coluna) que, acumulados durante a execução do processo, causam falhas por envelhecimento (quarta coluna).

Para exemplificar este encadeamento a primeira linha do quadro 3.1 será explicada em detalhes. Nesta, uma falha (ex. imperícia do programador) durante a codificação do software, introduziu uma falta (defeito) no código referente ao gerenciamento de alocação dinâmica de memória do processo servidor TELNET, embarcado em *switches* de rede[19]. Este defeito causa a liberação de apenas parte da memória alocada em rotinas relacionadas à criação e finalização de sessões TELNET. Como conseqüência, a cada ativação do código defeituoso (ex. abertura e encerramento de uma sessão TELNET) tem-se a perda de referências de porções de memória alocadas pelo processo, ocasionando o erro caracterizado como vazamento de memória. A ocorrência deste erro, repetidas vezes, causa o aumento do tamanho do processo servidor TELNET, reduzindo a quantidade de memória disponível no sistema para as outras tarefas. O efeito resultante do acúmulo deste erro é a exaustão da memória principal do sistema, o que acarreta na paralisação total do sistema por indisponibilidade deste recurso.

Para cada exemplo (linha) do quadro 3.1, na última coluna existem referências de casos reais que reportam situações similares às que estão sendo apresentadas. Vale ressaltar que diversas outras causas e efeitos estão relacionados às falhas por envelhecimento de software, contudo, procurou-se exemplificar apenas aqueles casos destacados com ênfase pela literatura.

[19] Em alusão ao caso real descrito em Cisco Systems (2000).

uma **Falha** de:	resulta em uma **Falta** de:	causando **Erro** de:	que provoca a **Falha**:
Codificação:	- Incorreta liberação de memória alocada dinamicamente.	a) Vazamento de memória.	Ex**:** Parada do sistema por insuficiência de memória principal. Caso: Cisco Catalyst (CISCO SYSTEMS, 2000).
Projeto:	- Valor de *timeout* inadequado para o perfil da aplicação.	b) Conexões pendentes.	Ex: Indisponibilidade de recursos para novas conexões. Caso: Problema de TCP FIN_WAIT no Tivoli Storage Manager (IBM, 2005).
Codificação:	- Encerramento de *threads* sem fechar conexões de rede e/ou banco de dados.		
Operação:	- Interrupção abrupta de conexões de rede e/ou banco de dados.		
Projeto:	- Tipificação incorreta de dados.	c) Perda de Precisão numérica.	Ex: Mudança incorreta de rota no sistema de navegação da aeronave. Caso: Ariane 5 (NEUMANN, 1995).
Codificação:	- Inconsistência na troca de valores entre variáveis (ex. inteiro ← real).		
Projeto:	- Não balanceamento na criação e remoção dos controles de compartilhamento de arquivos.	d) Não liberação de recursos de controle de concorrência.	Ex**:** Elevado tempo de resposta do processo e indisponibilidade de recursos de programação concorrente. Caso: Não liberação de semáforo por manipulador de exceção da rotina QUECALLS no OS/2 (IBM, 2000).
Codificação:	- Não desativação dos bloqueios (*lock*) de registros após seu uso.		
Projeto:	- Ausência de crítica em argumentos de funções susceptíveis ao estouro de pilha (*buffer overflow*).	e) Corrupção do estado volátil do processo.	Ex: Travamento ou falha geral do processo e/ou sistema operacional. Caso: Corrupção do *heap* do processo servidor RPC do IBM AIX 5.10 (IBM, 2004).
Codificação:	- Número de chamadas recursivas, de uma função, que excede um limite máximo permitido.		
Operação:	- Configuração de blocos ou setores incompatível para o perfil de operação exigido.	f) Fragmentação acentuada do sistema de arquivos.	Ex: Inconsistência na tabela de alocação do sistema de arquivos. Caso: Falha do processo *fsck* em corrigir as unidades de alocação de *inodes*. *Incident Number* 31948 (VERITAS, 2001).

Quadro 3.1: Exemplos de encadeamento (Falha➔Falta➔Erro➔Falha) relacionados a problemas de envelhecimento de software

3.2.2. Ocorrências

A natureza de ocorrência das FRE pode ser tanto do tipo *Bohrbug* quanto *Heisenbug* (VAIDYANATHAN; TRIVEDI, 2005). Como já observado no capítulo 1, faltas do tipo *Heisenbug*[20] são difíceis de serem detectadas, pois na maioria dos casos suas condições de ativação dependem do estado interno do processo, em especial do ambiente de sistema operacional e do perfil de operação (carga de trabalho) do software. Ambas dependências freqüentemente são variáveis aleatórias. Deste modo torna-se difícil reproduzir tais faltas nas fases de teste do software (*debugging*), resultando na sua propagação para as versões finais do produto de software. A figura 3.1 é uma extensão da taxonomia definida por Gray (1986), proposta em Vaidyanathan e Trivedi (2005).

Figura 3.1: Diagrama de Venn com a classificação de faltas de software por Trivedi
Fonte: Vaidyanathan e Trivedi (2005, p. 126)

A distinção entre FRE do tipo *Bohrbug* e FRE do tipo *Heisenbug*, ocorre de acordo com a natureza de ocorrência das falhas de envelhecimento que são conseqüências destas faltas. Por exemplo, uma falta de software que causa, de forma determinística, uma exaustão gradual de recursos é considerada uma FRE do tipo *Bohrbug*. O segundo cenário é uma falta causando um vazamento desconhecido de recursos durante ocorrências não determinísticas, sendo classificada como FRE do tipo *Heisenbug*.

Novamente tomando a primeira linha do quadro 3.1 como exemplo, a figura 3.2 apresenta dois fragmentos de código referentes a um servidor TELNET hipotético. Na figura 3.2(a), a falta se refere ao encerramento do processo/*thread* sem a liberação da memória alocada. Neste exemplo, esta condição sempre vai ocorrer quando a conexão for fechada pelo cliente TELNET. Nos casos onde este fechamento de conexões for considerado um padrão determinístico, então se trata de uma FRE do tipo *Bohrbug*. Já no segundo fragmento de código (ver figura 3.2(b)), quando a conexão é fechada pelo cliente antes do encerramento da sessão, ocorrem duas chamadas para a função *free*(3). A primeira chamada libera a memória previamente alocada com *malloc*(3), contudo a segunda chamada possui efeito indeterminado

[20] Na taxonomia de Avižienis *et al.* (2004) estas são equivalentes à classe de faltas intermitentes.

em muitos SO, podendo ou não causar um erro de vazamento de memória[21]. Isso ocorre porque seu efeito é dependente do estado volátil do processo, neste caso o *heap*[22]. Com isso, a reprodução do problema se torna de difícil execução, inclusive podendo ter seus resultados alterados caso o programa seja modificado para depuração. Tais características qualificam esta FRE como do tipo *Heisenbug*. A figura 3.3 apresenta os fluxos de execução dos códigos apresentados na figura 3.2.

```
…
int telnet_session(int opt)
{
 TELNET_CMD *ptr;
 …
 ptr=(TELNET_CMD *) malloc(TELNET_CMD_SIZE);
 for(;;) /* loop infinito */
 {
   ret=read(socket_fd, ptr, SIZE_CMD);
   if ( ret == -1 )
   {
     /* conexão fechada pelo cliente */
     return ret; /* encerra a função */
   }
   else
   {
     if ( strcmp(ptr,"exit") == 0 )
     {
       /* usuário deseja finalizar a sessão */
       break;
     }
     else
     {
       /*repassa comandos ao shell da sessão*/
     }
   }
 }
 free(ptr); /*libera memória alocada*/
 return ret;
}
```

(a)

```
…
int telnet_session(int opt)
{
 TELNET_CMD *ptr;
 …
 ptr=(TELNET_CMD *) malloc(TELNET_CMD_SIZE);
 for(;;) /* loop infinito */
 {
   ret=read(socket_fd, ptr, SIZE_CMD);
   if ( ret == -1 )
   {
     /* a conexão foi fechada pelo cliente */
     free(ptr); /*libera memória alocada*/
     break; /* encerra a função */
   }
   else
   {
     if ( strcmp(ptr,"exit") == 0 )
     {
       /* usuário deseja finalizar a sessão */
       break;
     }
     else
     {
       /*repassa comandos ao shell da sessão*/
     }
   }
 }
 free(ptr); /*libera memória alocada*/
 return ret;
}
```

(b)

Figura 3.2: Exemplo de faltas relacionadas ao envelhecimento dos tipos *Bohrbug* (a) e *Heisenbug* (b)

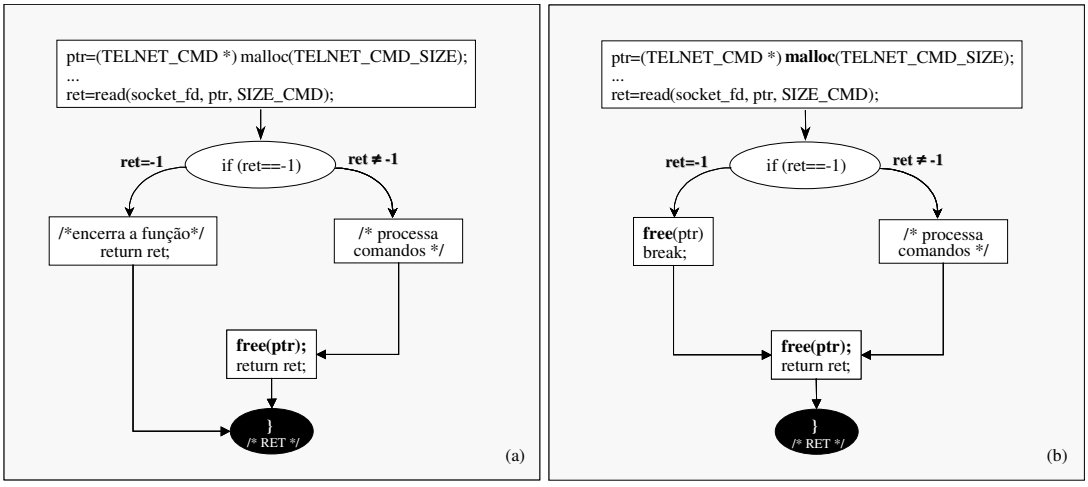

Figura 3.3: Ilustração dos fluxos de execução dos exemplos da figura 3.2

[21] No linux a chamada *free*(3) duas vezes possui efeito indeterminado segundo o manual *on-line* da função (*man page*). Neste exemplo, considera-se que este efeito pode causar um vazamento de memória.

[22] Região de memória responsável por armazenar objetos dinamicamente alocados.

As falhas por envelhecimento de software ocorrem como uma conseqüência do acúmulo dos efeitos resultantes das diversas ocorrências de FREs durante a execução de um processo. Na prática, mesmo naqueles casos onde a FRE é do tipo *Bohrbug*, a percepção da falha nem sempre é determinística. Tomando o exemplo ilustrado pelo fragmento de código da figura 3.2(a), verificou-se que existe uma relação determinística entre o fechamento da conexão e a ativação da falta. Contudo, a ocorrência da falha como conseqüência das diversas ativações desta falta dependerá de vários fatores, tais como o ambiente operacional (ex. quantidade de recursos de memória disponível) e a distribuição de freqüência das ativações das faltas, esta última sendo dependente do comportamento do usuário neste caso.

Portanto, ambos os tipos de FRE causam falhas por envelhecimento cuja natureza aleatória as tornam de difícil detecção e remoção. Devido a isso, as pesquisas nesta área têm se concentrado na modelagem do fenômeno do envelhecimento de software com vistas à sua melhor compreensão. Na próxima seção serão apresentados os principais trabalhos publicados nesta área.

3.3. MODELAGEM DO ENVELHECIMENTO DE SOFTWARE

Diversos trabalhos têm sido publicados propondo diferentes abordagens para a modelagem do envelhecimento de software. Basicamente, estes trabalhos se dividem em três enfoques: modelagem analítica, experimentação e estas duas abordagens integradas.

Independente da abordagem adotada, três fatores de grande importância para a captura do envelhecimento têm sido considerados nestes trabalhos (GARG, 1998b):

- efeitos do envelhecimento: os efeitos do envelhecimento têm sido observados principalmente no que concerne à sua percepção pelo usuário. Basicamente, dois casos são atualmente considerados: falha geral e degradação progressiva da performance;
- distribuição dos tempos de falha: atualmente não existe consenso com relação a um modelo teórico de probabilidade que caracterize os tempos de falhas de software durante sua fase operacional. Portanto, para que um modelo tenha ampla aplicabilidade é importante que este suporte várias distribuições, não se restringindo a uma em específico. Deste modo, a partir da disponibilidade de dados de campo ou experimentos, se pode utilizar a distribuição que melhor ajuste a estes dados;

- dependências: a maioria dos trabalhos na área (ver quadro 3.2) tem investigado a relação das falhas por envelhecimento com o tempo de execução do processo. Contudo, também existem estudos que, além do tempo de execução, consideram a carga de trabalho submetida ao sistema.

Além destes, outros aspectos também são importantes e devem ser considerados no estudo do envelhecimento, tais como o tempo de missão (execução completa do processo até cumprir seu objetivo) e a probabilidade de sucesso para se completar a missão em um dado intervalo de tempo. O quadro 3.2 apresenta uma comparação dos principais trabalhos que consideram as questões mencionadas anteriormente. Uma síntese destes trabalhos é apresentada na próxima seção.

Pesquisas	Categoria (Analít./Med.)	Env. capturado: (Falha/Degrad.)	Distribuições Gerais?	Dependência (Carga/Tempo)	Missão (Tempo/Prob.)
Huang *et al.* (1995)	Analítico	Falha	Não	Tempo	N/A
Garg *et al.* (1995a)	Analítico	Falha	Não	Tempo	N/A
Garg *et al.* (1995b)	Analítico	Falha	Não	Tempo	N/A
Garg *et al.* (1996b)	Analítico	Falha	Sim	Tempo	Tempo
Avritzer e Weyuker (1997)	Medição	Degradação	N/A	Carga e Tempo	N/A
Garg *et al.* (1998a)	Medição	Degradação	N/A	Tempo	Tempo
Garg *et al.* (1998b)	Analítico	Falha e Degradação	Sim	Carga e Tempo	N/A
Vaidyanathan e Trivedi (1999)	Medição	Degradação	Sim	Carga ou Tempo	Tempo
Trivedi *et al.* (2000)	Analítico e Medição	Falha e Degradação	Sim	Carga e Tempo	N/A
Bobbio, Sereno e Anglano (2001)	Analítico e Medição	Degradação	Sim	Tempo	Prob.
Castelli *et al.* (2001)	Analítico e Medição	Falha e Degradação	Sim	Tempo	N/A
Vaidyanathan e Trivedi (2001b)	Analítico	Falha e Degradação	Não	Tempo	N/A
Li,Vaidyanathan e Trivedi (2002)	Medição	Degradação	N/A	Tempo	N/A
Gross, Bhardwaj e Bickford (2002)	Medição	Degradação	N/A	Carga e Tempo	N/A
Shereshevsky *et al*. (2003)	Medição	Degradação	N/A	Carga e Tempo	N/A
Xie, Hong e Trivedi (2004)	Analítico	Falha	Não	Tempo	N/A
Vaidyanathan e Trivedi (2005)	Analítico e Medição	Falha e Degradação	Sim	Carga	Tempo
Bao, Sun e Trivedi (2005)	Analítico e Medição	Falha e Degradação	Sim	Carga e Tempo	N/A

Quadro 3.2: Principais aspectos abordados nas pesquisas em envelhecimento de software

3.3.1. Síntese das principais pesquisas na área

Diversos trabalhos têm-se baseado nos conceitos introduzidos por Huang *et al.* (1995). Este por ser o precursor nesta área será descrito em maior detalhamento.

Em Huang *et al.* (1995) foi definido o conceito de intervalo de longevidade base da aplicação, que se refere ao tempo desde a inicialização do processo até a sua entrada em um estado em que há a iminência de uma falha devido aos efeitos do envelhecimento de software. Em todo este intervalo se considera que a taxa de falha seja constante, basicamente próxima de zero, sendo representativo de um período mínimo de vida. A partir de seu modelo, Huang *et al.* objetivaram estimar tanto os custos de inatividade (*downtime*) programada (p/ manutenções preventivas), quanto não programada (devido à falhas por envelhecimento). A partir destas estimativas é possível avaliar a viabilidade em se realizar a manutenção preventiva do processo com sintomas de envelhecimento ou simplesmente assumir o risco de mantê-lo em execução sem paradas para manutenção.

A figura 3.4 representa um diagrama de transição de estados probabilístico definido por Huang *et al*. (1995), o qual apresenta os estágios relacionados à falha de um processo durante seu período de execução. De acordo com o modelo de Huang *et al*., quando o processo é criado ele se encontra em um estado robusto (S_0), o qual permanece pelo período do seu intervalo de longevidade base. A partir do acúmulo dos efeitos das faltas ativadas durante este período, o processo transita para um estado de iminência de falha (S_P) (*failure probable state*) a uma taxa (r_2) exponencialmente distribuída. A partir deste estado, a única transição possível é para S_F, que representa o estado de falha. A transição de S_P para S_F ocorre de acordo com uma taxa média (de falha) λ determinada. A transição do estado S_F para S_0, representa o reparo do processo após a falha, que pode ser tanto a criação de um novo processo quanto o acionamento de algum mecanismo de recuperação de falhas.

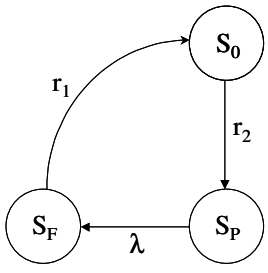

Figura 3.4: Modelo de estados de operação
Fonte: Huang *et al*. (1995)

No modelo de Huang *et al*., $1/r_2$ corresponde ao intervalo de longevidade base citado anteriormente. O tempo de reparo do processo segue uma taxa (r_1) exponencialmente distribuída, assim como a taxa de falha λ. A partir deste modelo definiu-se um valor de custo por *downtime*, onde é possível calcular este custo para um determinado intervalo de tempo *L* de acordo com a equação 3.1.

$$CustoDowntime(L) = \frac{1}{1+\frac{r_1}{\lambda}+\frac{r_1}{r_2}} \times L \times c_f \,, \tag{3.1}$$

onde:

L=intervalo de tempo que se deseja estimar o custo de *downtime*;

c_f=custo médio por unidade de tempo de *downtime* não programado.

A partir deste modelo, Huang *et al.* propõem a inserção de um estado que represente a reparação preventiva do processo, o qual é apresentado na figura 3.5. Como nas demais transições, as taxas r_3 e r_4 são exponencialmente distribuídas. Se o processo é reparado de forma preventiva, após *t* unidades de tempo, então $r_4=1/t$. A partir deste modelo, alguns cenários são simulados em Huang *et al*. (1995), o que permitiu aos autores concluírem que a freqüência de visitas ao estado S_R deve ser cuidadosamente programada, haja vista seu impacto nos custos de *downtime* para reparo. A quantidade de visitas ao estado S_R deve ser otimizada de forma que os custos de interrupção programada não excedam os benefícios de sua realização, ou seja, sendo inferiores aos custos de interrupções não programadas (falhas por envelhecimento).

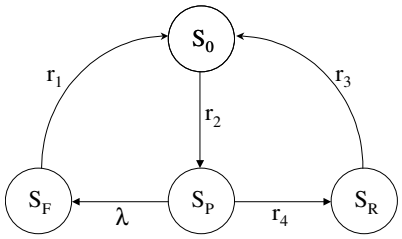

Figura 3.5: Modelo de transição com estado de reparo
Fonte: Huang *et al*. (1995)

No trabalho Huang *et al* (1995) foi adotado um modelo do tipo CTMC (*Continuous-Time Markov Chain Model*), diferentemente do trabalho de Garg *et al.* (1995b) que representou os estados do processo por uma Rede de Petri do tipo MRSPN (*Markov Regenerative Stochastic Petri Net*) (TRIVEDI, 2001). A principal diferença entre ambos

modelos é que na proposta de Garg *et al.* as manutenções preventivas ocorrem em intervalos fixos, iniciando a partir do estado robusto, ao contrário do modelo anterior em que estas somente ocorrem após o processo sair do estado robusto. Em termos de modelagem do envelhecimento os dois trabalhos representam a degradação do sistema através de um estado intermediário entre o estado robusto e o estado falho, denominado de estado de falha provável (S_P) (ver figura 3.5).

Seguindo o mesmo modelo de estados de Huang *et al.* (1995), o trabalho Garg *et al.* (1995a) visa minimizar a quantidade de solicitações que são perdidas ou rejeitadas em virtude da parada programada para manutenção preventiva do envelhecimento. Esta abordagem, além de oferecer uma maior granularidade para se calcular os custos de indisponibilidade do sistema, incluindo também os custos de perda por requisição durante o período de *downtime*, permite avaliar a melhor política naqueles casos onde os custos de parada não programada são os mesmos dos custos de paradas programadas, sendo estas as principais contribuições do trabalho.

Um outro enfoque foi dado em Garg *et al.* (1996a), que modelou o envelhecimento de software como o incremento na taxa de falhas em relação ao tempo de execução do sistema. Também, avaliou-se a utilização de *checkpoints* (JALOTE, 1994) e manutenções preventivas, com o objetivo de reduzir o tempo de missão do sistema que sofre de envelhecimento de software. O modelo proposto é genérico suficiente para acomodar qualquer tipo de distribuição de probabilidades para a taxa de falhas, contudo, os autores salientam a inadequação de algumas distribuições tal como a exponencial, já que o conceito de taxa de falhas constante se torna incompatível com as características de um software que sofre de envelhecimento de software. Para ilustrar numericamente sua proposta, os autores utilizaram a distribuição de Weibull que permitiu modelar os efeitos do envelhecimento de software a partir de uma taxa de falha incremental. Neste caso, o parâmetro de forma (*shape*) da distribuição deve ser maior que 1, haja vista que seu valor menor que 1 indica uma taxa de falha decrescente, bem como igual a 1 tem-se a distribuição de Weibull reduzindo-se a uma exponencial cuja taxa de falha é constante, sendo ambos casos incompatíveis com a caracterização do envelhecimento de software.

O primeiro trabalho que apresentou uma abordagem experimental para o envelhecimento de software, baseada na medição da degradação do sistema, foi Avritzer e Weyuker (1997). Este verificou que o desempenho de um grande sistema de telecomunicação decrementava monotonicamente com o seu tempo de execução. A diferença principal deste para os trabalhos já citados, em especial Huang *et al.* (1995), reside na utilização de medições

no sistema real a fim de se determinar os limiares de operação que antecedem os estados de falha ou de paradas para manutenção.

Avritzer e Weyuker coletaram amostras de tráfego de uma central telefônica em determinados momentos no tempo, juntamente com a mensuração da taxa de degradação do sistema, para se determinar a quantidade de pacotes perdidos em conseqüência da evolução dos efeitos degenerativos do envelhecimento de software. Esta atividade objetivou subsidiar a decisão de reinicializar ou não o sistema para o restabelecimento da sua capacidade nominal. Os autores enfatizam que esta tomada de decisão centrou-se basicamente na quantidade de pacotes que seriam perdidos durante o período de inatividade provocado pela reinicialização, em contraposição à perda de pacotes devido à degradação do sistema e/ou o período de restabelecimento após uma falha geral conseqüente da redução drástica de desempenho. Em especial, destaca-se no trabalho a abordagem orientada à degradação do sistema, em contrapartida à quantificação da taxa de falhas adotada nos trabalhos anteriores. Este enfoque ajusta-se bem ao tipo de fenômeno estudado, o qual normalmente implica em uma evolução da degradação da performance de um ou mais recursos devido o envelhecimento de software.

Como resultado, Avritzer e Weyuker validaram sua metodologia através de um modelo linear, representando a evolução da degradação. Para isso, os dados utilizados foram amostrados durante períodos de 20 minutos de tráfego diário durante um ciclo que durou nove meses. A partir destes dados, foi possível estimar o limiar para se realizar a interrupção do sistema, em detrimento de mantê-lo em operação de forma degradada.

Em Garg *et al.* (1998b), um modelo estocástico foi definido para representar um software do tipo servidor, no qual as transações chegam a uma taxa constante e são enfileiradas e atendidas seguindo uma disciplina FCFS (*First-Come-First-Served*). A taxa de atendimento é uma função qualquer, sendo os efeitos do envelhecimento de software modelados utilizando tanto uma taxa de serviço decrescente quanto uma taxa de falhas crescente. Ambas condições podendo ser em função do tempo, da carga instantânea do sistema, da carga acumulada média ou de uma combinação destas. A degradação na capacidade de atendimento das transações, bem como as falhas do sistema, são modelados como processos estocásticos independentes. Neste trabalho, Garg *et al.* inovam em possibilitar a modelagem tanto da degradação do sistema quanto das ocorrências de falhas, ambas conseqüências do envelhecimento, permitindo que tais efeitos sejam em função do tempo de execução como também da carga de trabalho (*workload*). Esta possibilidade de parametrização oferece uma grande flexibilidade para a avaliação dos efeitos do envelhecimento em sistemas servidores com as características especificadas anteriormente.

Com base em um enfoque experimental baseado em medição, diferentemente das abordagens analíticas apresentadas anteriormente, com exceção de Avritzer e Weyuker (1997), Garg *et al.* (1998a) apresenta uma metodologia para a detecção e estimação do envelhecimento de software baseada na coleta e análise de dados do ambiente operacional. Os autores utilizam como instrumentação para a coleta de dados, agentes SNMP (*Simple Network Management Protocol*) (STALLINGS, 1998) distribuídos em diversas estações de trabalho UNIX, os quais monitoram variáveis definidas em uma MIB (*Management Information Base*) criada pelos autores para este propósito. São exemplos de variáveis desta MIB: a quantidade de memória principal disponível, ocupação da partição de *swap*, estados dos processos, estatísticas de operações de E/S, dentre outras. Os autores definiram um período de coleta de 53 dias, com a monitoração ocorrendo a cada 15 minutos. A partir das séries de dados obtidas, a metodologia inicialmente avalia se existe algum tipo de tendência nos dados que indique a degradação de recursos do sistema. Para os casos onde existam efeitos sazonais nos dados, o trabalho recomenda a utilização do teste de Kendall para sazonalidade (*seasonal Kendall test*) (GILBERT, 1987 apud GARG *et al.,* 1998a). Os autores verificaram a existência de uma tendência linear nos dados e obtiveram seu coeficiente angular a partir de um método não-paramétrico denominado *Sen's slope* (TRIVEDI, 2001, p. 744). Este procedimento é adotado sob a justificativa de que a inclinação estimada por mínimos quadrados, pelo método de regressão linear, é susceptível a maiores desvios na presença de valores extremos (*outliers*), ao contrário do método proposto. A partir da equação da reta obtida, os tempos de exaustão dos recursos analisados foram estimados.

No trabalho descrito anteriormente, os autores modelaram o efeito do envelhecimento de software, com base na tendência de degradação detectada na utilização dos recursos do sistema. Por exemplo, verificou-se uma tendência decrescente na quantidade de memória principal disponível com o decorrer do tempo de operação ininterrupta do sistema. Havendo a ausência de tendência de degradação na série de dados de um dado recurso, significa que não existe a ocorrência dos efeitos do envelhecimento de software naquele recurso. Os próprios autores destacam, como principal limitação da sua metodologia, a predição do tempo de falha em função da exaustão de apenas um único recurso de cada vez. Eles sugerem a extensão do trabalho, no sentido de se considerar a interação e a correlação entre a degradação de vários recursos e seus impactos na disponibilidade do sistema avaliado como um todo.

Uma evolução do trabalho anterior se apresenta em Vaidyanathan e Trivedi (1999), o qual propõe um enfoque, também baseado em medição, para a estimação da exaustão de recursos como conseqüência do envelhecimento de software. Neste foi considerada a

tendência de exaustão de recursos não somente em função do tempo, mas também da carga de trabalho do sistema. Para tanto, um modelo semi-Markov (SMP) foi construído a partir dos dados da carga de trabalho e utilização de recursos coletados de estações UNIX usadas na pesquisa. Em termos de coleta de dados foi utilizado o mesmo procedimento descrito em Garg *et al.* (1998a). A caracterização das cargas de trabalho considerou variáveis referentes às atividades relacionadas à CPU e operações de E/S no sistema de arquivos. A partir deste conjunto de variáveis, um número de estados representando diferentes cargas de trabalho foi identificado, juntamente com a probabilidade de transição destes estados e a distribuição do tempo de permanência em cada estado (*sojourn time*), valores estes necessários para a construção do modelo SMP.

A identificação dos estados se deu através da análise estatística de agrupamentos (*cluster analysis*), estratificando os diversos estados referentes às diferentes cargas de trabalho às quais os sistemas monitorados foram submetidos. Para a análise de agrupamentos os autores utilizaram um algoritmo não hierárquico iterativo denominado *Hartigan's k-means* (HARTIGAN, 1975 apud VAIDYANATHAN; TRIVEDI, 1999). As distribuições *hiper-exponencial* e *hipo-exponencial* (TRIVEDI, 2001), foram usadas para descrever o tempo de permanência (*sojourn time*) nos estados. Suas aderências aos dados obtidos foram testadas, utilizando o teste K-S (*Kolmogorov-Smirnov*) (TRIVEDI, 2001) em um nível de significância (α) igual a 0,01, onde ficaram comprovadas suas adequações.

O efeito do envelhecimento foi modelado utilizando-se os dados obtidos na etapa de medição. Os mesmos procedimentos de Garg *et al.* (1998a) foram adotados, ou seja, buscou-se a existência de tendência nos dados e, ao ser comprovada, calculou-se o coeficiente angular de inclinação da reta de regressão. Para cada estado do modelo calculou-se um coeficiente. Os dados usados no cálculo pertenceram aos mesmos intervalos de amostragem usados para a caracterização das cargas de trabalho. Para um estado que representa determinada carga de trabalho, foi possível verificar a taxa de consumo/exaustão de um recurso durante diferentes visitas àquele estado, as quais foram relativamente próximas. Em contrapartida, entre os estados estes valores foram diferentes, o que validou a suposição inicial dos autores de que a carga de trabalho tem relação direta com o envelhecimento de software. Verificou-se também que quanto maior a carga de trabalho, maior foi o consumo/degradação de recursos. Complementarmente, os autores verificaram que as estimativas de exaustão dos recursos, baseadas no modelo que considerou a carga de trabalho, apresentaram uma melhor acuracidade do que aquelas baseadas apenas no tempo de execução.

Em Trivedi, Vaidyanathan e Goseva-Popstojanova (2000) realizou-se basicamente uma síntese do modelo analítico proposto por Garg *et al.* (1998b) e da abordagem baseada em medição proposta em Vaidyanathan e Trivedi (1999). Avaliando as conclusões dos autores, para ambas abordagens, verifica-se em comum a forte relação entre a carga de trabalho e o envelhecimento. Na modelagem analítica, destaca-se a verificação dos autores que desde que um software não estando em execução (estado ocioso na memória), este não se encontra susceptível ao envelhecimento. Portanto, um cenário mais realista é a modelagem da taxa de falha e de degradação do serviço como uma função do tempo de processamento (CPU *time*), e não do tempo total de execução (*calendar time*), já que neste último existem períodos de atividade e inatividade (*idle time*).

Bobbio, Sereno e Anglano (2001) introduziram o conceito de índice de degradação em seu modelo analítico. Os autores assumiram que o valor deste índice pode ser obtido a partir da medição de determinados recursos do sistema. A evolução da degradação, observada através do índice de degradação, é modelada como um processo estocástico. Em termos da modelagem do envelhecimento, considerando seus efeitos a partir da evolução gradual da degradação, trabalhos anteriores tais como Garg *et al.* (1998b) têm desenvolvido uma formulação similar.

A contribuição do modelo proposto por Bobbio, Sereno e Anglano reside na maior granularidade da especificação da degradação. Os autores consideram duas hipóteses para a captura da evolução do envelhecimento (acúmulo do índice de degradação). A primeira diz respeito à possibilidade de se monitorar o índice de degradação em intervalos eqüidistantes no tempo. Nesta, a degradação do sistema é medida em intervalos fixos no tempo (figura 3.6(a)), sendo que o valor de incremento (X_i) do índice de degradação é considerado uma variável aleatória cuja distribuição de probabilidades deve ser inferida experimentalmente do sistema real. A equação 3.2 representa esta proposição.

$$s(k\Delta t) = h_k = S_{\min} + \sum_{i=1}^{k} X_i \,, \tag{3.2}$$

onde:

s(t) é o índice de degradação monitorado no tempo t;

Δt é o intervalo de observação;

k representa o intervalo monitorado;

h_k representa a degradação acumulada até o k-ésimo intervalo;

S_{min} é valor de s(t) medido quando o sistema está totalmente disponível, no início de sua fase operacional;

X_i é o valor de incremento do índice de degradação medido na i-ésima observação.

Já na segunda hipótese, com relação à captura do acúmulo da degradação, considera-se o incremento no índice de degradação uma função do número de faltas que ocorrem em um determinado intervalo de tempo ($s(t)$). Neste caso, o tempo entre as ocorrências de faltas (X_i) e o valor de incremento do índice por ocorrência (Y_i) são considerados variáveis aleatórias que seguem distribuições de probabilidades que devem ser inferidas a partir de dados obtidos experimentalmente. Deste modo, Bobbio, Sereno e Anglano destacam que o processo de degradação é completamente especificado se a probabilidade $P_k(t)$, de ter k faltas no tempo t, é conhecida. As figuras 3.6(a) e 3.6(b) sintetizam ambas propostas de Bobbio, Sereno e Anglano (2001). A ocorrência de uma falha acontece quando o nível de degradação ultrapassa um limiar máximo tolerável pré-estabelecido para o sistema/processo observado.

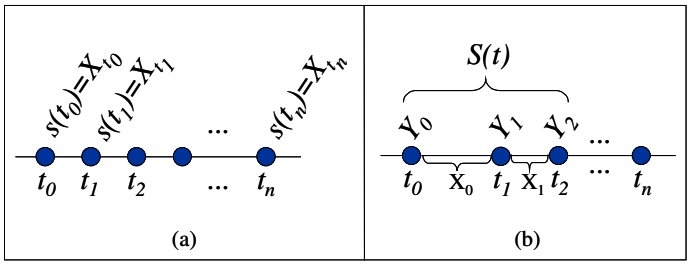

Figura 3.6: (a) Modelagem da captura do acúmulo de degradação em intervalos eqüidistantes e (b) pelo número de faltas por intervalo de tempo

Em termos de aplicação da modelagem do envelhecimento de software, o trabalho Castelli *et al.* (2001) foi pioneiro ao integrar ambas abordagens (modelagem analítica e experimentação) em uma aplicação real voltada para a manutenção preventiva de sistemas de software servidores susceptíveis ao envelhecimento. Esta tecnologia atualmente é utilizada pela IBM em seu produto denominado IBM *Director®*. Um dos módulos do *Director* é encarregado de monitorar os processos de aplicação e recursos do sistema operacional, de forma a estimar os tempos de exaustão destes recursos a fim de gerar alertas para a infra-estrutura de gerenciamento pró-ativo de falhas. Estes alertas são emitidos quando o nível de consumo dos recursos ultrapassa horizontes pré-definidos pelo usuário. A figura 3.7 apresenta exemplos das interfaces do software.

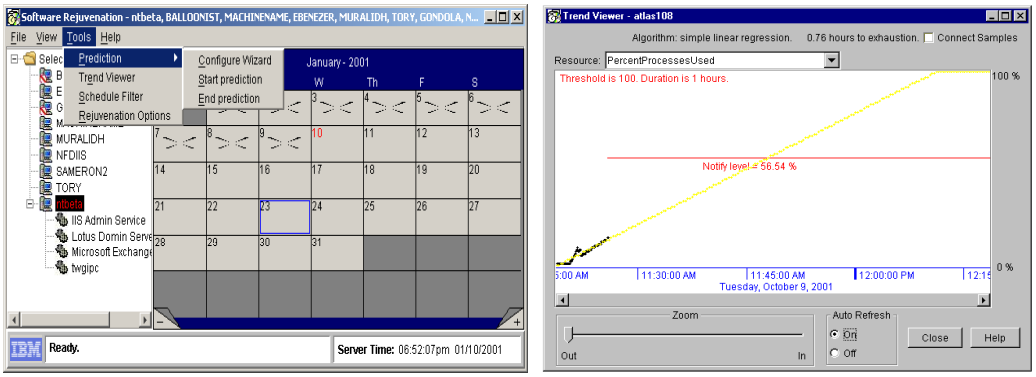

Figura 3.7: Módulos principal (esq.) e de análise de tendência do IBM *Director*
Fonte: Castelli *et al.* (2001)

A metodologia proposta em Castelli *et al.* (2001) considera as seguintes etapas: (1) pré-processamento das amostras obtidas durante a monitoração, (2) teste de aderência de vários modelos aos dados resultantes do passo anterior, (3) seleção do modelo mais adequado e (4) previsão do tempo de falha.

No passo (2) os autores utilizam vários modelos (ex. regressão linear simples e com *2* pontos de parada) e para avaliar a acuracidade dos modelos utilizou-se o índice C_p de Mallows (DRAPER; SMITH, 1981 apud CASTELLI *et al.,* 2001; MALLOWS, 2000). A partir do modelo selecionado, são realizadas extrapolações de forma a se ter estimativas sobre o comportamento futuro da exaustão dos recursos monitorados.

O trabalho de Vaidyanathan *et al.* (2001c) é basicamente uma extensão da pesquisa citada anteriormente, contudo seu enfoque é para a modelagem do sistema para fins de manutenção preventiva contra os efeitos do envelhecimento. Este utilizou uma variação das redes de Petri estocásticas, chamada SRN (*stochastic reward nets*), para modelar um sistema de aglomerados (*cluster*) de servidores. Este segue uma abordagem de modelagem do envelhecimento similar àquela proposta por Huang *et al.* (1995), onde o envelhecimento é caracterizado pelo incremento na taxa de falha, representado pela transição de um estado robusto para um estado de falha provável. O diferencial do trabalho reside em sua aplicação voltada para *clusters* de servidores, onde a falha do sistema é caracterizada apenas quando todos os nodos estão no estado de falha. A utilização de uma política de manutenção preventiva com base no modelo de predição proposto, considerando um *cluster* com um único sistema sobressalente (máximo de dois nodos), reduziu o *downtime* em 62% em relação ao cenário sem manutenção.

Em Li, Vaidyanathan e Trivedi. (2002) um estudo avaliando os efeitos do envelhecimento de software em servidores web foi realizado. A partir da geração de cargas de

trabalho sintéticas, os autores avaliaram a existência ou não de tendência na exaustão de recursos do servidor web Apache (THE APACHE GROUP, 2005). A tendência dos dados foi verificada para cada variável monitorada (ex. tempo de resposta do *webserver*, memória física disponível, outras). O enfoque adotado foi o mesmo usado nos trabalhos anteriores que se basearam na medição da degradação dos recursos, em especial Castelli *et al.* (2001), Garg *et al.* (1998a) e Avritzer e Weyuker (1997).

Depois de confirmada a existência de tendência nos dados, Li, Vaidyanathan e Trivedi utilizaram técnicas de análise de séries temporais, em especial ARMA (*Autoregressive Moving Average*) (HANKE; REITSCH; WICHERN, 2001), com o objetivo de detectar e quantificar o envelhecimento dos processos do Apache. Dentre as limitações do trabalho, os autores destacam a dependência do seu modelo com relação à periodicidade em que seus parâmetros devem ser re-estimados, juntamente com a necessidade de uma melhor investigação da relação entre a degradação da performance do servidor e os níveis de exaustão dos recursos monitorados.

Shereshevsky *et al.* (2003) faz uma crítica ao modelo linear proposto em Vaidyanathan e Trivedi (1999), tomando por base as altas flutuações nos dados reportados por aqueles autores. Segundo Shereshevsky *et al.*, a grande variabilidade nos dados não é compatível com a suposição de tendência linear assumida pelos autores, resultando em limites de confiança muito amplos, o que torna proibitiva sua utilização. Shereshevsky *et al.* observam que a dinâmica dos recursos monitorados naquele trabalho poderia ser mais bem representada por modelos não lineares. Neste sentido, Shereshevsky *et al.* (2003) propõem uma aplicação da teoria de fractais para modelar os efeitos do envelhecimento na forma de degradação dos recursos do sistema operacional. Com base nesta abordagem, duas hipóteses foram formuladas: (1) a dinâmica temporal dos parâmetros relacionados à memória do sistema operacional tem uma estrutura multifractal e (2) a análise dos padrões multifractais nas séries temporais destes parâmetros pode ser usada para determinar quando o sistema operacional está envelhecendo e, portanto, estando susceptível a uma falha por envelhecimento de software.

Como principal ferramenta quantitativa Shereshevsky *et al.* (2003) usam o expoente de Hölder, o qual mede a intensidade das oscilações de uma função na vizinhança de um determinado ponto onde a função é definida (DAOUDI *et al.,* 1998 apud SHERESHEVSKY *et al.*, 2003). A computação e análise deste expoente são amplamente utilizadas para se estudar a estrutura fractal de séries temporais (SHERESHEVSKY *et al.,* 2003). Os valores do expoente (na faixa de 0 até 1) determinam o grau de "fractalidade" de uma função, sendo que

quanto mais próximo de 0 significa que a oscilação da série se intensifica, suavizando quando os valores se aproximam de 1.

Para a etapa experimental, Shereshevsky *et al.* (2003) selecionaram dentre as diversas variáveis do sistema operacional[23], relacionadas à utilização da memória, duas que melhor demonstraram possuírem características *multifractais*, que foram: *freemem* (número médio de páginas de memória disponíveis para processos do usuário) e *lg_alloc* (quantidade de memória alocada para atender grandes requisições dos processos). A figura 3.8 apresenta as séries de dados referentes à variável *freemem* e os respectivos valores do expoente de Hölder. Como se pode observar, o intervalo marcado em ambos os gráficos ilustra a sensibilidade do expoente de Hölder para com a instabilidade dos dados referentes ao parâmetro *freemem* (gráfico superior).

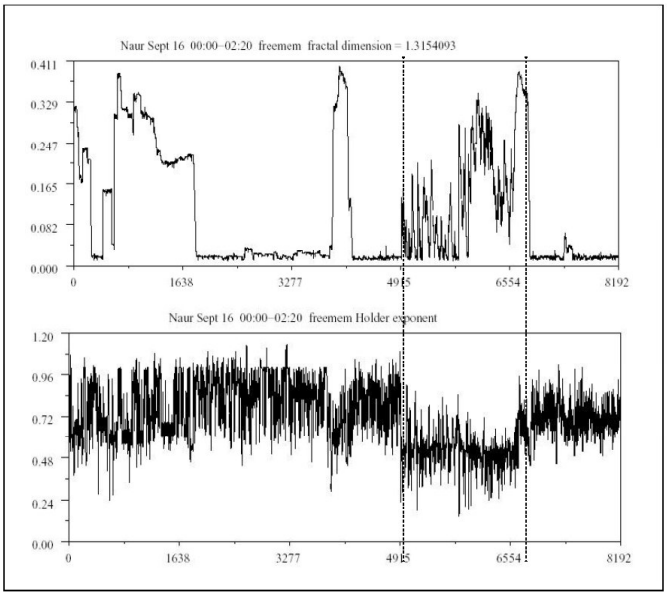

Figura 3.8: Séries dos valores do parâmetro *freemem* e do seu expoente de Hölder
Fonte: Shereshevsky *et al.* (2003)

A partir de diversos experimentos que levaram o sistema a uma falha por exaustão de recursos, utilizando cargas de trabalho elevadas, os autores verificaram que em 80% dos casos a falha do sistema foi precedida por dois decrementos no expoente de Hölder. Os autores argumentam que tais evidências reforçam sua hipótese de que este expoente pode ser representativo de um estado de degradação que antecede falhas por envelhecimento. Ressalta-se que o estudo contemplou apenas cargas de trabalho que foram incrementadas com o tempo,

[23] Os experimentos foram realizadas em servidores UNIX utilizando o sistema operacional SunOS 5.5.1 e estas variáveis foram coletadas através do programa *sar* (*system activity report*) que acompanha este sistema.

não tendo sido avaliadas alternâncias (incremento e decremento) nestas cargas, a fim de verificar se o padrão de comportamento, apresentado pela série temporal do expoente de Hölder, se mantém também para este cenário.

Um estudo também voltado para a monitoração em tempo real da degradação de recursos do sistema, como forma de detectar e predizer os efeitos do envelhecimento de software, foi conduzido em Gross, Bhardwaj e Bickford (2002). Neste, avaliou-se a aplicabilidade do método MSET (*Multivariate State Estimation Technique*) para a detecção do envelhecimento de software. O MSET (GROSS; WEGERICH; SINGER, 1998 apud GROSS; BHARDWAJ; BICKFORD, 2002) é uma técnica de modelagem estatística, não paramétrica e não linear, utilizada para o monitoramento de faltas que ocorrem em instalações na área de energia nuclear. Esta se baseia em um conjunto de dados de treinamento, o qual caracteriza a operação normal do sistema observado. Posteriormente, aplica-se a técnica SPRT (*Sequential Probability Ratio Test*) (HUMENIK; GROSS, 1990 apud GROSS; BHARDWAJ; BICKFORD, 2002) para se determinar quando os desvios dos valores observados, em relação ao padrão de operação pré-estabelecido, não são característicos da operação normal do sistema. Não sendo desvios característicos da operação padrão, tem-se um indicativo da presença de faltas no ambiente monitorado. Diversas variáveis relacionadas a recursos internos do sistema operacional (hardware e processos de aplicações) foram monitoradas, sendo que ao final do período de amostragem foram selecionadas 50 variáveis para treinamento e teste do modelo MSET. Na etapa experimental, Gross, Bhardwaj e Bickford (2002) utilizaram um injetor de faltas para provocar erros de vazamento de memória. Desta forma, foi possível a replicação controlada do experimento para a avaliação da sensibilidade do MSET aos efeitos do envelhecimento produzidos sinteticamente. Como resultado, a técnica se mostrou bastante acurada em termos de sua sensibilidade aos desvios de comportamento dos recursos face aos efeitos do envelhecimento, tendo exibido 100% de acerto na emissão dos alarmes indicando a presença do envelhecimento. Os autores salientam que esta técnica, tipicamente, exibe entre 1% a 2% de alarmes falsos, o que é extremamente positivo em termos de ambientes de missão crítica, onde a interrupção do serviço em detrimento de tais alarmes possui custos bastante elevados.

Em Vaidyanathan e Gross (2003) foi realizado um estudo complementar ao de Gross, Bhardwaj e Bickford (2002), onde se fez uma análise de sensibilidade do MSET, considerando o número de vetores de treinamento para a caracterização do padrão de operação normal do sistema, bem como o número de variáveis consideradas no modelo e o tempo de teste necessário para treinamento.

Uma abordagem complementar ao trabalho de Vaidyanathan *et al.* (2001c) é apresentada em Xie, Hong e Trivedi (2004). Esta utilizou o mesmo paradigma de SRN utilizado no trabalho Vaidyanathan *et al.*, contudo considerou que o tempo de permanência no estado robusto, bem como o MTTF, dependem da carga de trabalho e não apenas do tempo de execução. Em termos de modelagem, os autores definiram dois níveis para a carga de trabalho (*peak hours* e *off-peak_hours*). A partir destes níveis, suposições foram admitidas com relação aos valores de diversos parâmetros do modelo, em especial dos tempos de permanência e MTTF, que estão diretamente ligados à modelagem do envelhecimento de software. Como exemplo das suposições feitas pelos autores, os seguintes tempos médios foram assumidos: permanência no estado "robusto" (4 horas c/ *workload=peak* e 3 dias c/ *workload=off-peak*) e MTTF (20 horas c/ *workload=peak* e 3 dias c/ *workload=off-peak*). Tais valores não foram baseados em nenhum estudo experimental, tendo sido utilizados apenas para a validação numérica do modelo.

Vaidyanathan e Trivedi (2005) apresenta um enfoque sistemático para a modelagem do envelhecimento e a predição do tempo de exaustão de recursos do sistema operacional, bem como a otimização do escalonamento de ações preventivas para mitigar os efeitos do envelhecimento de software, visando prevenir falhas e interrupções não programadas. Nesta proposta, Vaidyanathan e Trivedi integram as duas principais abordagens atuais (analítica e baseada em medições) voltadas para a modelagem do envelhecimento. Esta metodologia, ilustrada na figura 3.9, teve como base seus vários trabalhos anteriores, principalmente Garg *et al.* (1998a) e Vaidyanathan e Trivedi (1999, 2001a, 2001b).

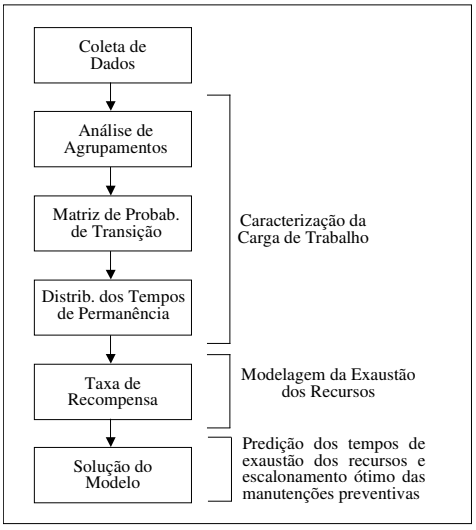

Figura 3.9: Enfoque para modelagem e predição dos efeitos do envelhecimento de software
Fonte: Traduzido de Vaidyanathan e Trivedi (2005)

O primeiro processo (coleta de dados) do modelo apresentado na figura 3.9, refere-se à etapa experimental da metodologia proposta por Vaidyanathan e Trivedi (2005), estando as etapas subseqüentes dependentes de sua saída. Portanto, o tempo necessário para realizar a coleta de dados é um fator determinante para a viabilidade daquele enfoque, haja vista que o elevado período de observação/experimentação poderia inviabilizar a sua aplicação. Neste cenário aplica-se a abordagem proposta neste estudo, cujo objetivo é a redução do tempo da etapa experimental realizando a aceleração do envelhecimento de software.

Em Bao, Sun e Trivedi (2005), foi apresentado um modelo hierárquico para análise do gerenciamento pró-ativo de faltas (PFM) na presença de vazamentos de recursos (*resource leakage*) do sistema. O modelo apresenta duas camadas, onde na camada de baixo nível se tem a modelagem do processo de degradação implementada através de uma cadeia de Markov, a qual estabelece a conexão entre o vazamento de recursos e a taxa da falha. Já na camada de alto nível se encontra o PFM modelado, o qual tem como entrada a análise de degradação da camada de baixo nível. Em termos do modelo de degradação da performance, considerou-se este em função da perda de recursos, com especial ênfase para o vazamento de memória (*memory leak*). O modelo suporta análises de ambos cenários, ou seja, com e sem o vazamento de recursos, avaliando a taxa de falha em cada caso.

A configuração com vazamento de memória é modelada através de uma cadeia de Markov do tipo CTMC. A carga de trabalho é caracterizada como um fluxo de requisições de recursos, o número de requisições em trânsito no sistema e o tamanho das requisições. Deste modo, o modelo de envelhecimento proposto considerou a carga de trabalho como um fator de influência na evolução da degradação de recursos. De acordo com a configuração do modelo, as chegadas de requisição são provenientes de uma fonte geradora de Poisson. A existência ou não de vazamentos de recursos é representada por uma função $l(t)$, a qual é não decrescente no tempo t para os casos onde se tem o vazamento de memória, bem como assume o valor zero quando se deseja representar a ausência de vazamento (*leak-free case*). O coeficiente $\xi[k,l]$ representa a probabilidade condicional de que o sistema falhe no estado k, caso chegue uma nova requisição e o total de memória indisponível devido ao vazamento seja l. O mapeamento do vazamento de memória e da carga de trabalho ocorre através dos parâmetros k e l, onde k é o número de processos de aplicação mantendo porções de memória alocadas, e l é o total de memória perdida devido ao vazamento (*leaked resource*). Neste caso, considera-se $l=l(t)$. Resolvendo analiticamente o modelo, os autores comprovaram que na presença de vazamento de memória, a taxa de falha incrementou monotonicamente na medida que o montante de memória indisponível devido ao vazamento se acumulou. No cenário onde

$l(t)$=0 (sem vazamento de memória) a taxa de falha se manteve constante. Portanto, o trabalho valida analiticamente a suposição, realizada por diversos trabalhos anteriores, contudo sem demonstração formal, de que os efeitos do envelhecimento de software provocados por vazamentos de recursos (não apenas memória) são caracterizados pelo incremento monotônico da taxa de falhas.

3.3.2. Principais críticas aos trabalhos apresentados

O objetivo da seção anterior foi apresentar uma revisão dos principais trabalhos de pesquisa realizados nos últimos anos, à cerca do problema do envelhecimento de software, com enfoque em suas abordagens tanto de forma analítica quanto experimental.

No contexto da abordagem analítica, os trabalhos focam na modelagem do envelhecimento estabelecendo sua relação tanto com a taxa de falha, quanto com a carga de trabalho. A limitação da maioria destes trabalhos reside nas suposições adotadas para os diversos parâmetros considerados em seus modelos, as quais não são baseadas em uma teoria ou em observação, de forma a explicar os valores utilizados. Como conseqüência, estes trabalhos não oferecem conclusões práticas a respeito do fenômeno do envelhecimento de software aplicado a situações reais.

Alternativamente à modelagem analítica, têm-se as pesquisas experimentais voltadas para um enfoque baseado em medições, onde os efeitos do envelhecimento são avaliados empiricamente ou experimentalmente. Nestes casos, a ênfase dada tem sido no sentido de verificar a presença ou não do envelhecimento. Além da detecção dos efeitos do envelhecimento, complementarmente têm-se buscado modelos para a predição dos tempos de falha resultantes dos efeitos degenerativos do envelhecimento de software, como por exemplo, a degradação da memória devido aos erros de vazamento (*memory leaks*). Neste contexto, a crítica é primeiramente direcionada aos trabalhos que têm como objetivo a detecção do envelhecimento.

Estes trabalhos não apresentam critérios claros para se declarar quando o sistema está ou não sofrendo do problema do envelhecimento de software. Em todos os trabalhos a conclusão de que o sistema está envelhecendo é normalmente baseada na tendência de degradação observada, subsidiada pelos resultados obtidos através dos diversos métodos (ex. *Sen's slope*, regressão linear) adotados. Ainda não existe um padrão de limiar para que o nível de degradação possa ou não ser considerado um sintoma do envelhecimento, ficando a cargo do analista esta decisão.

Uma segunda crítica com relação aos trabalhos experimentais, voltados para a detecção e mensuração do envelhecimento, está na orientação voltada aos efeitos do envelhecimento, desconsiderando suas causas. Nenhuma das pesquisas referenciadas investigou as causas (fatores) que influenciaram no envelhecimento dos sistemas analisados. Entende-se que, na maior parte destes trabalhos, o principal objetivo foi buscar modelos de predição dos tempos de falha por envelhecimento, justificando a maior ênfase nos efeitos do envelhecimento. Contudo, a caracterização do fenômeno do envelhecimento, a partir dos fatores que mais influenciam sua ocorrência, permite não só melhorar a compreensão do problema, como também obter modelos de predição a partir do perfil de operação dos processos, em função dos fatores que ativam e/ou intensificam seu envelhecimento.

3.3.3. Contribuição desta pesquisa para a literatura atual

Como se pode observar na evolução dos trabalhos apresentados, a obtenção de dados sobre o envelhecimento, de forma experimental, a fim de subsidiar os modelos analíticos, tem sido uma tendência nesta área de pesquisa. Contudo, a obtenção destes dados é dificultada não só pela natureza estocástica do fenômeno do envelhecimento de software, como também pelo crescente incremento nos níveis de confiabilidade e robustez dos sistemas.

Quando se trata de sistemas altamente confiáveis, a dificuldade em se obter uma amostra de falhas é ainda maior, haja vista que a degradação dos processos por envelhecimento normalmente ocorre lentamente, sendo percebida somente após um longo período de execução ininterrupta do software. Esta problemática fica evidenciada nos tempos de experimentação dos trabalhos com enfoque em mediação. Durante a revisão da literatura, realizou-se um levantamento dos tempos de experimentação daqueles trabalhos que possuíam esta informação ou que fosse possível computá-la, os quais estão listados no quadro 3.3.

Trabalhos Experimentais	Tempo necessário para a detecção dos efeitos do envelhecimento de software
Huang *et al*. (1995)	14 dias
Avritzer e Weyuker (1997)	270 dias
Trivedi, Vaidyanathan e Goseva-Popstojanova (2000)	53 dias
Bobbio, Sereno e Anglano (2001)	82 dias
Li, Vaidyanathan e Trivedi (2002)	46 dias
Shereshevsky *et al*. (2003)	> 120 dias
Wang *et al*. (2004)	27 dias
Vaidyanathan e Trivedi (2005)	> 90 dias

Quadro 3.3: Exemplos de tempo de experimentação em envelhecimento de software

Ressalta-se que a maioria destes trabalhos envolveu experimentos cujo objetivo foi apenas identificar a presença ou não do envelhecimento de software, não tendo sido observado o sistema até a sua falha, o que exigiria uma maior tempo de observação. Além disso, salienta-se que nenhum destes experimentos considerou a replicação das medições, sendo estes tempos equivalentes a um único período de observação.

Neste contexto, o presente trabalho tem por objetivo contribuir para o campo da engenharia de software experimental, em especial engenharia de confiabilidade de software, apresentando um enfoque sistematizado para acelerar a ocorrência de falhas por envelhecimento de software e, conseqüentemente, reduzir o tempo (custo) de experimentação nesta área de pesquisa.

3.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado um estudo detalhado sobre os principais aspectos relacionados ao envelhecimento de software. As principais publicações nesta área foram descritas, enfatizando os procedimentos utilizados pelas duas principais abordagens adotadas atualmente: modelagem analítica e estudos experimentais voltados para a detecção e mensuração dos efeitos do envelhecimento de software. A integração de ambas abordagens em um terceiro enfoque, tido como híbrido, tem se mostrado uma tendência nesta área.

Os tempos de duração dos experimentos, reportados pelos trabalhos revisados, apóiam a justificativa de se desenvolver métodos que permitam obter tempos de falha por envelhecimento de software com a redução do período de experimentação. Vale lembrar, que muitos dos trabalhos citados não apresentam os tempos de falha, mas sim os tempos utilizados para se detectar os sintomas do envelhecimento. Isto significa que períodos de experimentação, superiores aos reportados, seriam necessários para a observação das falhas por envelhecimento de software.

CAPÍTULO 4

ENSAIOS ACELERADOS

4.1. INTRODUÇÃO

O atual nível de competitividade, nas diversas áreas da indústria, tem imposto aos fabricantes o desafio de desenvolver novos produtos com menor tempo e custo, mantendo ou preferencialmente ampliando seus níveis de confiabilidade em campo. Em termos práticos, isso significa que as informações referentes à confiabilidade (dados de vida) destes produtos devem ser obtidas em um período de tempo aceitável para que possam ser usadas na melhoria destes produtos, bem como dos novos projetos (FREITAS; COLOSIMO, 1997).

Tradicionalmente, a análise de dados de vida envolve avaliar os tempos de falha, obtidos sob condições normais de operação do produto, de maneira a quantificar as suas características de vida (METTAS, 2003). Em muitas situações, e por diversas razões, tais dados são difíceis, se não impossíveis, de serem obtidos em tempo hábil para sua utilização. Segundo Mettas (2003), as razões para esta dificuldade podem incluir os elevados níveis de confiabilidade apresentados pelos atuais produtos, em resposta à crescente demanda do mercado por padrões de qualidade cada vez maiores. Também, outro aspecto a se considerar é o curto espaço de tempo disponível entre o projeto e a entrega do produto.

Em virtude destes desafios, a indústria e em especial a área de engenharia de confiabilidade, têm desenvolvido métodos para obter os tempos de falha mais rapidamente do que se pode alcançar em condições normais de uso, ou seja, estes métodos visam à aceleração da ocorrência de falhas (METTAS, 2003). Nos últimos anos, a principal abordagem adotada para este propósito tem sido os testes acelerados (*accelerated testing*) (NELSON, 2004). Este termo representa uma variedade de métodos que envolvem a aceleração de falhas de produtos/sistemas com o propósito de quantificar as suas características de vida (METTAS, 2003). A partir dos dados de tempos de falha obtidos por estes métodos de aceleração, pode-se realizar a análise da confiabilidade que normalmente não seria possível visto as dificuldades citadas anteriormente.

Na seqüência serão apresentados os principais conceitos de ensaios acelerados que serão usados para implementar o método proposto no capítulo 5.

4.2. ASPECTOS CONCEITUAIS

Basicamente, os ensaios acelerados podem ser classificados em três grupos, os quais são: testes qualitativos, testes de seleção e testes quantitativos.

Os testes qualitativos têm por objetivo informar ou produzir informações à cerca dos modos de falha do sistema/produto sob teste (SST). Estes testes têm sido referenciados por várias nomenclaturas, incluindo *elephant tests*, *torture tests*, HALT (*highly accelerated life tests*) e *shake & bake tests* (METTAS, 2003). Normalmente, estes testes são realizados com pequenas amostras, onde os espécimes são submetidos a um único e severo nível de estresse, a vários estresses, ou estresses não constantes no tempo (*time-varying stress*). Caso o espécime sobreviva aos testes, o mesmo está aprovado. Em caso contrário, ações apropriadas serão realizadas para melhorar o projeto, de forma a eliminar a(s) causa(s) da falha. Portanto, testes qualitativos são usados principalmente para revelar modos de falha (METTAS, 2003). Estes testes não produzem dados que permitem caracterizar a vida (ou a confiabilidade) do sistema sob teste.

Os testes de seleção (*environment stress screening* - ESS) implicam na utilização de algum tipo de estímulo ambiental aplicado ao SST de forma acelerada. Estes estímulos podem ser variações térmicas, estresse elétrico, vibrações, dentre outros. O objetivo destes testes é expor, identificar e eliminar defeitos latentes, os quais não poderiam ser eliminados por inspeção visual ou testes elétricos (ex. testes de "energização"), mas causariam falhas durante a operação em campo. Estes testes são executados sobre toda a população e não envolvem amostragem (METTAS, 2003). Um caso especial de ESS é o teste de *Burn-in*, o qual também é executado para seleção e eliminação de dispositivos com defeitos de fabricação ("aberrações de produção"), os quais possuem falhas dependentes de estresse e tempo (METTAS, 2003). Normalmente, estes testes são aplicados para componentes elétricos e eletrônicos. Aqueles componentes que sobreviverem após um determinado período de tempo provavelmente não apresentarão falhas prematuras em sua operação, tendo em vista que os mesmos já foram expostos a esta etapa durante o próprio processo de fabricação (NELSON, 2004). Tanto o ESS quanto o *Burn-in*, assim como os testes qualitativos, não produzem dados para quantificar a confiabilidade.

Os testes acelerados quantitativos, diferentemente dos anteriores, são projetados para quantificar as características de vida dos produtos em suas condições normais de uso, o que conseqüentemente resulta na produção de informações que propiciem a sua análise de confiabilidade. Os testes acelerados utilizados neste trabalho são do tipo quantitativos, haja

vista seu propósito principal de produzir dados de vida que permitam a posterior análise de confiabilidade dos sistemas que estão sendo investigados. As próximas seções deste capítulo abordarão os principais conceitos envolvendo os testes acelerados quantitativos.

4.2.1. Ensaios acelerados quantitativos

De forma geral, os ensaios acelerados quantitativos podem ser divididos em dois tipos. Esta classificação é realizada em função da característica dos dados coletados (FREITAS; COLOSIMO, 1997):

- Ensaios de vida acelerados (***accelerated life tests*** - ALT): a resposta de interesse é o tempo até a ocorrência da falha. Nestes testes estimam-se os tempos médios entre/até as falhas (MTBF/MTTF), a função confiabilidade, dentre outras medidas de interesse baseadas em condições de operação do SST que propiciem a aceleração da ocorrência de suas falhas. Ressalta-se que estas condições de aceleração diferem do padrão de uso normal do SST. Com base na modelagem dos dados obtidos em condições de aceleração, são calculadas as medidas de confiabilidade de interesse, que devem ser extrapoladas para as condições normais de uso (sem aceleração) do SST.
- Ensaios de degradação acelerados (*accelerated degradation tests* – ADT): neste caso a resposta de interesse é alguma medida de performance do SST, obtida durante os testes realizados em condições de aceleração das falhas. A partir dos dados de degradação estima-se a distribuição dos tempos de falha. Contudo, esta distribuição é obtida para as condições de aceleração, o que exige a utilização dos mesmos procedimentos de extrapolação utilizados nos ensaios de vida acelerados.

4.2.2. Variáveis de estresse

Independente do tipo de ensaio acelerado (ALT ou ADT), é fundamental a definição de como será realizada a aceleração das falhas. Para tanto, o conceito de variável de estresse é essencial. Uma variável de estresse, ou também chamada de estresse de aceleração (*accelerating stress*), é aquela que quando utilizada em níveis diferentes daqueles adotados em condições normais de uso, reduz o tempo até a falha do produto/sistema que está sendo testado (FREITAS; COLOSIMO, 1997).

Para muitos produtos/materiais existem padrões de variáveis de aceleração. Por exemplo, alta temperatura e voltagem são normalmente utilizadas em ensaios acelerados para testar isolamento elétrico e também dispositivos eletrônicos (NELSON, 2004). A maioria destes padrões tem como base conhecimentos em áreas de engenharia de materiais, elétrica e eletrônica, de alimentos, química, dentre outras. Porém, existem casos onde padrões de variáveis de estresse ainda não estão bem estabelecidos, ficando a cargo do experimentador a definição das variáveis de estresse a serem utilizadas. Como exemplo, os produtos de software se enquadram nesta categoria. Nestes casos, um estudo experimental se faz necessário, onde serão definidas as variáveis de estresse que aceleram os modos de falha de interesse (NELSON, 2004). Os quadros 4.1 e 4.2 apresentam alguns exemplos de variáveis de estresse e medidas de performance já estabelecidas por tipo de materiais e produtos.

Materiais		
Tipo	**Medida de Performance**	**Variável de Estresse**
Metais	Trinca; corrosão; oxidação	Temperatura; umidade; sal
Dielétricos e isolamentos	Tempo até a falha; alongamento	Temperatura; voltagem; vibração
Alimentos e drogas	Tempo de estocagem; pH; reações químicas específicas	Temperatura; umidade; radiação solar
Plásticos	Propriedades mecânicas; firmeza da cor	Temperatura; vibração; choque

Quadro 4.1: Materiais, medidas de performance e variáveis de estresse
Fonte: Freitas e Colosimo (1997)

Produtos		
Tipo	**Medida de Performance**	**Variável de Estresse**
Semicondutores e componentes microeletrônicos	Tempo até a falha e características de operação	Temperatura; corrente; voltagem; umidade; pressão
Capacitores	Tempo até a falha	Temperatura; voltagem; vibração.
Resistores	Tempo até a falha	Temperatura; voltagem; vibração
Contatos elétricos	Corrosão, tempo até a falha.	Temperatura; umidade; corrente
Lâmpadas	Tempo até a falha, eficiência, luminosidade	Voltagem; temperatura; choque (elétrico ou mecânico)

Quadro 4.2: Produtos, medidas de performance e variáveis de estresse
Fonte: Freitas e Colosimo (1997)

A seleção cuidadosa das variáveis de estresse, bem como dos seus níveis de intensidade, se faz necessária, a fim de evitar a ocorrência de modos de falha que não se manifestariam nas condições de projeto, ou seja, no padrão de operação normal do SST.

4.2.3. Métodos de aceleração

Tanto para ALT quanto ADT, a aceleração das falhas ocorre submetendo o SST a níveis mais altos de estresse do que aqueles encontrados nas suas condições normais de uso (FREITAS; COLOSIMO, 1997). Isto é realizado adotando níveis elevados para as variáveis de estresse, em comparação aos níveis de uso. Em Nelson (2004), os dois principais métodos de aceleração são apresentados, os quais serão descritos a seguir.

4.2.3.1. Aceleração por elevação da taxa de uso

Uma forma de acelerar a vida de muitos produtos é colocá-los em operação utilizando uma taxa de uso mais elevada do que o normal. Duas alternativas para se implementar este método de aceleração são (NELSON, 2004):

- Aumentar a velocidade de operação: neste caso, coloca-se o SST em funcionamento a uma velocidade mais alta do que ele opera em condições de não aceleração. Por exemplo, um motor poderia ser colocado em funcionamento a uma velocidade superior a que foi definida como seu padrão de operação.
- Redução no tempo de descanso: Muitos produtos se encontram desligados uma grande parte do seu período de vida. Por exemplo, em muitas residências a máquina de lavar roupas não opera mais do que 1 hora por dia. Nestes casos, o SST é colocado em operação por uma fração de tempo maior. No exemplo da máquina de lavar roupas, ela poderia operar em um regime de 12 ou 24 horas ininterruptas. Ressalta-se que, diferente do método anterior, a velocidade de operação é a mesma daquela em regime normal de uso, tendo sido incrementado apenas a fração de tempo de operação sem interrupção.

De acordo com Meeker e Escobar (1998), uma suposição fundamental deve ser assegurada para a validade dos testes de aceleração baseados na taxa de uso. Esta estabelece que a vida útil do SST, representada pela distribuição de ciclos até a falha, não deve sofrer alterações em conseqüência do tipo de ciclo de operação utilizado, nem da taxa ou freqüência adotadas para estes ciclos nos níveis acelerados. Ou seja, se em condições normais de operação (ex. operação durante 1 hora/dia) o MTBF da máquina de lavar roupas é de 10.000 ciclos, em condições de estresse acelerado (ex. operação de 20 horas/dia) o MTBF deve ser o mesmo do regime normal de operação, ou seja, 10.000 ciclos.

4.2.3.2. Aceleração por altos níveis de estresse (overstress)

No enfoque anterior a variável de estresse a ser controlada foi a taxa de uso. Porém, em muitos casos o estresse de aceleração é uma variável que, se utilizada em níveis superiores àquele definido para sua condição normal de operação, causará uma redução no tempo de vida do SST ou uma maior degradação de sua performance (NELSON, 2004). Alguns exemplos de variáveis de estresse utilizadas para este fim estão listados nos quadros 4.1 e 4.2.

4.2.4. Formas de aplicação da carga de estresse

As formas de aplicação da carga de estresse, que serão apresentadas nesta seção, estão relacionadas apenas ao método de aceleração por altos níveis de estresse.

Em Mettas (2003) são definidos dois grupos, denominados estresse constante ou independente do tempo (*constant stress/time-independent*) e estresse variável ou dependente do tempo (*varying stress/time-dependent*). De acordo com Nelson (2004), a escolha da forma de aplicação da carga de estresse depende de como o SST trabalha em campo, bem como de restrições teóricas e práticas envolvidas. Também, este mesmo autor salienta que as teorias que analisam os efeitos do estresse variável, sobre a vida de produtos/sistemas, ainda são rudimentares, sendo que muitas destas não têm sido verificadas apropriadamente.

Em função dos motivos citados anteriormente, Nelson (2004) ressalta que em alguns casos é preferível utilizar estresse constante, mesmo quando o SST experimenta estresse variável em campo/serviço. Complementando sua justificativa, o referido autor destaca que a aplicação de estresse constante é mais fácil de ser implementada e, para muitos casos onde o estresse manifesta-se de forma variável, é possível a sua adequada representação utilizando estresse constante.

4.2.4.1. Estresse constante

O estresse constante ou independente do tempo é o tipo mais comum de forma de aplicação de estresse em um ensaio acelerado. O SST é submetido a um mesmo nível de estresse que se mantém durante todo teste. A fim de realizar a modelagem adequada dos dados obtidos neste tipo de teste, de acordo com o método de ensaios acelerados, se faz necessário utilizar vários níveis de estresse constante.

A figura 4.1 ilustra esta forma de aplicação. Neste exemplo, foram testados 12 espécimes, sendo alocadas 4 unidades para cada nível de estresse. No nível mais alto de estresse todos os espécimes falharam antes do término do teste. No nível intermediário três espécimes falharam e um sobreviveu. No menor nível, dois espécimes falharam e dois sobreviveram.

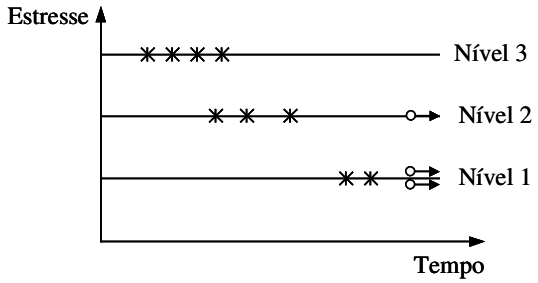

Figura 4.1: Tipo de teste com estresse constante

A definição da duração do teste (eixo x) é fundamental para se verificar se os espécimes testados têm falhado ou não durante o teste, sendo um dos aspectos que serão abordados na seção 4.3.1.

4.2.4.2. Estresse variável

Neste tipo de teste o SST é submetido a um nível de estresse que muda com o tempo. Neste enfoque, espera-se que o SST apresente falhas mais rapidamente do que quando aplicado estresse constante (METTAS, 2003). Esta forma de aplicação pode ser conduzida de acordo com as seguintes abordagens:

- Escada (*step-stress*): Cada SST é submetido a um nível de estresse por um período de tempo (figura 4.2(a)). Se o SST não falhar neste período, o nível de estresse é elevado para um novo patamar e o procedimento se repete. A principal vantagem desta forma de aplicação de estresse é levar o SST à manifestação de falhas mais rapidamente. Freitas e Colosimo (1997) salientam que a maior desvantagem deste enfoque está nas estimações das medidas de confiabilidade a partir dos dados de vida (tempos de falha) obtidos. Os autores destacam que a maioria dos produtos em condições normais de uso é submetida a uma carga constante e não do tipo "escada". Neste caso, o modelo a ser adotado é mais complexo, pois deve considerar o efeito

cumulativo da exposição do SST aos vários níveis de estresse sucessivos (NELSON, 2004). Além disso, Freitas e Colosimo (1997) lembram que os modos de falha que ocorrem nos patamares mais altos, em geral, podem diferir daqueles que ocorrem em condições normais de uso.

- Progressivo: Cada SST é submetido a um nível crescente de estresse, porém esse aumento não é feito em "passos" como no enfoque anterior, mas sim progressivamente (figura 4.2(b)). Os testes progressivos possuem as mesmas vantagens e desvantagens dos testes em "escada". Contudo, nestes pode ser difícil o controle acurado do nível de estresse durante o ensaio acelerado (NELSON, 2004).

- Cíclico: Cada SST é submetido a estresses de níveis alto e baixo de forma cíclica (figura 4.2(c)). Segundo Nelson (2004), para muitos produtos os ciclos consideram os mesmos níveis de estresse no nível mais alto do ciclo. Em outros casos, os mesmos níveis de estresse nos níveis alto e baixo se repetem até a falha. Nestes casos, a vida do SST é mensurada em números de ciclos até a falha.

- Aleatório: Existem casos onde o SST está sujeito a níveis de estresse que se alteram de maneira aleatória (figura 4.2(d)). Como exemplo, Nelson (2004) cita componentes estruturais de pontes e de aviões. Nestes casos os testes empregam estresses aleatórios seguindo a mesma distribuição de variação que ocorre durante a operação do SST, porém usando valores mais elevados para os níveis de estresse. Nelson (2004) destaca que tal como nos testes cíclicos, testes aleatórios possuem características (ex. média e desvio padrão) que permitem simplificar sua execução e análise através de uma abordagem baseada em estresse constante. Por exemplo, adotando-se um valor médio para o nível de estresse, ou até mesmo um valor superior ao valor médio, o teste poderia ser conduzido na forma de estresse constante e os dados modelados seguindo esta abordagem. Corroborando com Nelson (2004), Freitas e Colosimo (1997, p. 159) destacam que:

> "[...] quando o modo de falha e o efeito da aceleração da variável de estresse são bem entendidos, e quando se deseja fazer estimativas para um ambiente cujo estresse é aproximadamente constante, utilizar testes com outra forma de aplicação de estresse que não a constante significa complicar desnecessariamente a situação".

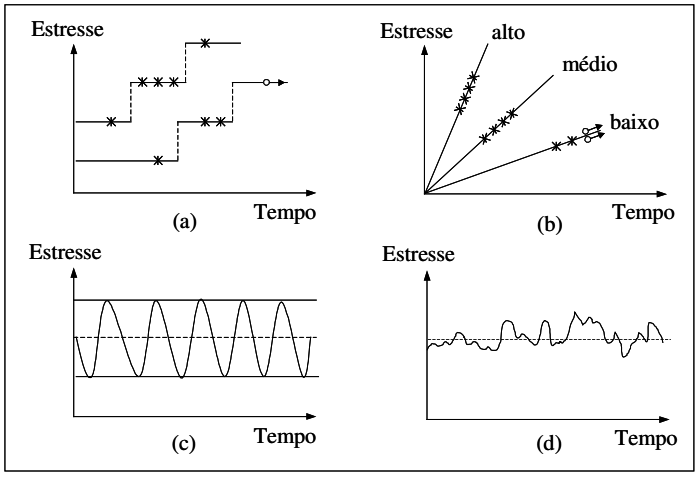

Figura 4.2: Variantes de teste de estresse dependente do tempo
Fontes: Nelson (2004)

4.2.5. Modelo de aceleração para ALT

O modelo que será apresentado considera os casos em que se tem uma única variável de estresse, apenas um modo de falha e a forma de aplicação do estresse é constante. Em Nelson (2004) e Meeker e Escobar (1998) podem ser encontrados modelos para mais de uma variável de estresse, múltiplos modos de falha e formas não constantes de aplicação do estresse.

Segundo Meeker e Escobar (1998), a interpretação dos dados de ensaios acelerados quantitativos exige modelos que relacionem as variáveis de estresse, nos seus diferentes níveis, com a aceleração do tempo de falha. De forma geral, estes modelos consistem de duas partes: uma distribuição de vida que caracteriza a variabilidade dos tempos de falha em cada nível de estresse, e um modelo de relacionamento vida-estresse que mapeia a maneira como esta distribuição se projeta entre os diferentes níveis de estresse (METTAS, 2003).

Com base nestes dois elementos se realiza a predição dos parâmetros da distribuição de vida que será usada para modelar os tempos de falha do SST no nível de uso (FREITAS; COLOSIMO, 1997; MEEKER; ESCOBAR, 1998; METTAS, 2003). A figura 4.3 ilustra um comportamento típico da distribuição de vida entre dois níveis da variável de estresse. Neste caso, tem-se a distribuição para um nível alto de estresse e foi realizada a sua projeção para o nível de uso. Na medida que o nível de estresse aumenta é esperado que o tempo de vida seja menor. Na próxima seção serão apresentados os principais conceitos envolvidos neste tipo de análise.

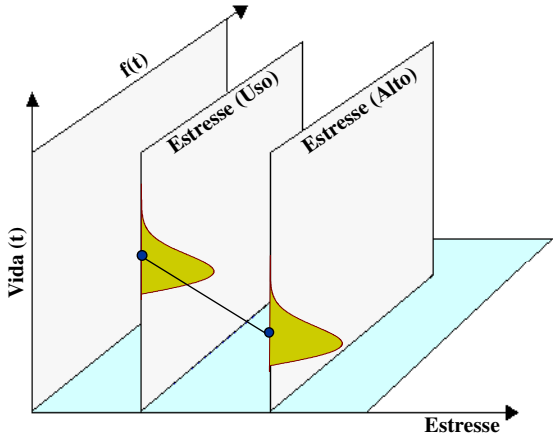

Figura 4.3: Distribuição de vida em dois níveis de estresse

4.2.5.1. Forma geral da relação vida-estresse

O relacionamento vida-estresse tem como objetivo estabelecer a relação entre o tempo de falha e a variável de estresse. Este modelo pode assumir diversas formas, tais como linear, exponencial, logarítmica, dentre outras.

Normalmente estes modelos são estabelecidos com base em leis e fenômenos físico-químicos, os quais permitem descrever a relação entre os efeitos que determinadas variáveis de estresse têm sobre o tempo de vida do material, produto, reação ou processo químico. Alguns modelos de relacionamento vida-estresse, tais como os exemplos citados no quadro 4.3, já se encontram bem estabelecidos em áreas onde ALT tem sido extensivamente aplicado.

Tipo de estresse de aceleração	Modelo de relacionamento
Temperatura	Arrhenius
Temperatura p/ degradações químicas	Eyring
Fadiga de materiais sujeitos a ciclos térmicos	Coffin-Manson
Não-térmicos	Inverse Power Law (IPL)
Temperatura-humidade	Peck
Temperatura-estresse elástico	Zhurkov

Quadro 4.3: Exemplos de modelos de relacionamento vida-estresse
Fontes: Nelson (2004) e Freitas e Colosimo (1997)

Para os principais tipos de estresse já se tem bem estabelecido um conjunto de modelos de relacionamento (NELSON, 2004; FREITAS; COLOSIMO, 1997). Em muitos casos (ex. IPL) é possível se derivar outros modelos especializados para a análise de determinadas aplicações das variáveis de estresse. Uma representação genérica do modelo de relacionamento vida-estresse é a equação 4.1.

$$tempo = f\left(estresse\right), \tag{4.1}$$

onde a forma funcional de *f* deve ser tal que expresse a tendência decrescente do *tempo de vida* em relação ao *estresse*.

Este relacionamento pode ser proveniente tanto de modelos físicos quanto empíricos (MEEKER; ESCOBAR, 1998). No primeiro caso, tem-se uma teoria explicando a relação entre o mecanismo de falha e a variável de estresse, como por exemplo, os modelos do quadro 4.3. O segundo caso aplica-se, alternativamente, quando existe pouco entendimento do processo que leva à falha, também denominado "física da falha". Nestes casos, os modelos são obtidos a partir da observação do processo ou também por meio de experimentação.

A figura 4.4(a) apresenta um exemplo de relacionamento vida-estresse na forma de uma curva exponencial. De acordo com Nelson (2004), a análise dos dados provenientes de ALT pode ser facilitada quando são utilizadas escalas alternativas como, por exemplo, a escala logarítmica. Neste caso, os valores normalmente assumem uma tendência linear e a reta traçada representa o relacionamento vida-estresse. Segundo o mesmo autor, a vantagem desta abordagem é que a análise dos dados na forma "linearizada" (figura 4.4(b)), torna-se matematicamente menos complexa do que quando se tem a representação em forma de curvas.

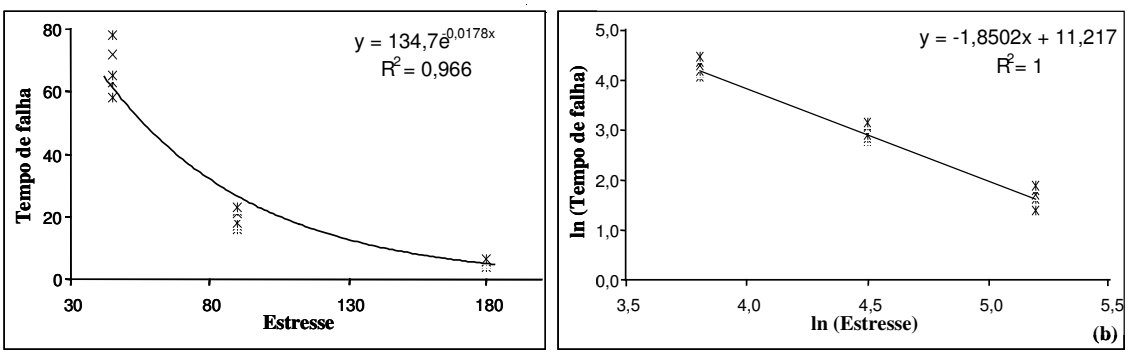

Figura 4.4: (a) Modelo de relacionamento vida-estresse; (b) Modelo de relacionamento transformado para a forma linear

Como exemplo de um modelo de relacionamento vida-estresse, a seguir será apresentada a relação de potência inversa (IPL) (ver quadro 4.3) juntamente com sua aplicação através de um exemplo numérico. Esta relação será utilizada no capítulo 5, pelo método proposto, para modelar a aceleração do envelhecimento de software.

4.2.5.1.1 Relação de potência inversa (IPL)

Esta relação é recomendada para modelar o tempo de falha em função de qualquer tipo de variável de estresse que seja não termal e assuma valores positivos. De acordo com Freitas e Colosimo (1997), alguns exemplos de aplicação da IPL incluem testes de vida acelerados com lâmpadas incandescentes, fadiga de materiais, isolantes, dielétricos, etc. Em Nelson (2004, p. 85), outros exemplos de aplicação e referências a trabalhos utilizando este modelo são apresentados. A equação 4.2 mostra uma das representações utilizadas para a relação IPL:

$$\tau(V) = \frac{1}{K \cdot V^n}, \tag{4.2}$$

onde:

τ = vida característica ou nominal (ex. vida média/MTTF);

V = nível de estresse;

K ($K > 0$) e n são parâmetros característicos do SST, método de fabricação e geometria, método de teste, dentre outros, e devem ser determinados com base nos dados (NELSON, 2004).

A forma "linearizada" da equação 4.2 está representada em (4.3).

$$\ln(\tau) = -\ln(K) - n \cdot \ln(V) = \beta_0 + \beta_1 x, \tag{4.3}$$

onde $\beta_0 = -\ln(K)$, $\beta_1 = -n$, e $x = \ln(V)$.

Nesta forma, a estimação dos parâmetros K e n é realizada encontrando-se o intercepto e o coeficiente de inclinação da equação 4.3, o que permite obter o valor de τ para o nível de estresse de interesse. A fim de exemplificar estes procedimentos, o mesmo conjunto de dados usados para construir a figura 4.4(b) foi utilizado. Estes dados foram obtidos de Reliasoft (2001) e se referem a um estudo de caso em ALT, onde a variável de estresse é não termal e positiva, indicando a possibilidade de aplicação do relacionamento IPL. O tempo considerado é o número de ciclos até a falha por fadiga de um produto sendo submetido a um determinado tipo de tensão mecânica. Com base na equação da reta da figura 4.4(b), os valores de K e n são calculados a seguir:

$\ln(\tau) = -\ln(K) - n \cdot \ln(V) = 11{,}217 - 1{,}8502 \cdot \ln(V)$,

$-\ln(K) = 11{,}217$, portanto $\hat{K} = \exp(-11{,}217) \cong 1{,}34\text{E-}05$

$\hat{n}$ = -(-1,8502) = 1,8502.

A partir dos valores estimados para o modelo IPL, pode-se projetar uma estimativa do tempo de falha em um nível de estresse de interesse. Por exemplo, para o nível de estresse 190 o tempo de vida obtido foi de aproximadamente 4,5 ciclos, como demonstra a expressão a seguir:

$$\hat{\tau}(V) = \frac{1}{\hat{K} \cdot V^{\hat{n}}} = \frac{1}{0{,}000013 \cdot 190^{1{,}8502}} \cong 4{,}522$$

A relação vida-estresse é baseada em um modelo determinístico e, portanto, representa apenas um valor pontual para cada nível da variável de estresse. Esta limitação não lhe permite capturar a variabilidade dos dados da variável resposta (tempo de falha).

Levando em conta estas limitações, em conjunto com o modelo de relacionamento vida-estresse, faz-se necessário à utilização de uma distribuição de probabilidades para descrever a variabilidade dos tempos de falha em cada nível de estresse, o que será descrito a seguir.

4.2.5.2. Distribuição de vida e o relacionamento IPL

A análise de dados de vida de ensaios acelerados exige, além do relacionamento vida-estresse, um componente probabilístico que permita modelar a variabilidade existente nos tempos de falha da amostra. Este componente segue uma certa distribuição de probabilidades e independe da variável de estresse, estando representado na equação 4.4 pelo termo "ε".

$$tempo = f(estresse) + \varepsilon \tag{4.4}$$

Deste modo, o tempo de falha depende do valor do estresse e segue a mesma distribuição de probabilidades que a variável aleatória ε. De forma simplificada, (4.4) pode ser visto como um modelo de regressão linear simples (DRAPER; SMITH, 1981), onde a variável dependente é o *tempo* e o *estresse* sendo a variável preditora.

Dentre as estratégias para analisar dados provenientes de ALT, tanto Meeker e Escobar (1998, p. 495) quanto Freitas e Colosimo (1997, p. 185), adotam um enfoque baseado em modelos de regressão para distribuições da família localização-escala ou que possam ser transformadas (log-localização-escala) para esta forma. Esta família de distribuições engloba as principais distribuições usadas em estudos de ALT, a saber: Exponencial, Normal, Weibull,

Lognormal, Logistic, Loglogistic e distribuições de valor extremo (MEEKER; ESCOBAR, 1998, p. 78).

Estas distribuições possuem uma propriedade importante para a análise de dados de ensaios acelerados, que diz respeito com a forma da sua função de distribuição acumulada (CDF). Segundo Meeker e Escobar (1998, p. 78), uma variável aleatória Y é proveniente desta família de distribuições se a sua CDF pode ser expressa como (4.5).

$$Pr(Y \leq y) = F(y;\mu,\sigma) = \Phi\left(\frac{y-\mu}{\sigma}\right) \tag{4.5}$$

Substituições apropriadas, demonstradas no capítulo 4 de Meeker e Escobar (1998), mostram que Φ é a CDF de Y quando μ=0 e σ=1 e que Φ é a CDF de $(Y - \mu) / \sigma$. Em casos onde a variável aleatória T procede de distribuições da família log-localização-escala, $Y = \log(T)$ é membro da família localização-escala e o modelo anterior se aplica. Para a parametrização apresentada em (4.5), diz-se que $-\infty < \mu < \infty$ é um parâmetro de localização e $\sigma > 0$ é um parâmetro de escala.

Segundo Freitas e Colosimo (1997, p. 185), a importância dos modelos provenientes das famílias (log-)localização-escala está em se assumir que o parâmetro de localização, na parametrização da CDF em (4.5), depende da variável de estresse x, ou seja $\mu(x)$, e que o parâmetro de escala seja independente de x. Esta descrição pode ser representada como:

$$Y = \log(T) = \mu(x) + \sigma\varepsilon \tag{4.6}$$

Existem várias opções para a forma funcional da dependência de μ na variável de estresse x. Segundo Freitas e Colosimo (1997, p. 185) a forma mais simples e utilizada é a linear. Assumindo um relacionamento linear, nota-se que (4.6) toma a seguinte forma:

$$Y = \log(T) = \beta_0 + \beta_1 x + \sigma\varepsilon \tag{4.7}$$

Este é um modelo de regressão linear simples quando $Y = \log(T)$ tem distribuição normal, com média (parâmetro de localização) $\mu(x) = \beta_0 + \beta_1 x$ e variância (parâmetro de escala) σ^2 ou, de forma equivalente, quando ε tem distribuição normal com média zero e variância igual a 1. A forma geral do modelo de aceleração usado neste trabalho, para analisar os dados obtidos com a aceleração do envelhecimento, será aquela representada por (4.7). Os livros Freitas e Colosimo (1997), Meeker e Escobar (1998) e Nelson (2004), discutem de

forma detalhada os aspectos teóricos que suportam o estabelecimento dos modelos de regressão para as famílias de distribuições citadas anteriormente.

Após selecionar a distribuição de vida e o modelo de relacionamento vida-estresse, assume-se que o parâmetro da distribuição que dependerá da variável de estresse é aquele que caracteriza a vida média do SST. Isto pode ser claramente visto no modelo 4.7, onde $\beta_0 + \beta_1 x$ representam o valor médio de $Y = \log(T)$. Este parâmetro é denominado por alguns autores (METTAS, 2003; Nelson, 2004) de parâmetro de vida característica ou ponto característico.

O quadro 4.4 apresenta os parâmetros de vida característica para as distribuições Weibull, Exponencial e Lognormal, que segundo Nelson (2004) são as mais usadas em ALT. Vale observar que no caso da distribuição Weibull, o parâmetro η na parametrização usada em (4.5) assume o papel de parâmetro de localização (μ).

Distribuição	Parâmetros	Vida característica
Weibull	β, η	Parâmetro de escala (η)
Exponencial	λ	Vida média ($1/\lambda$)
Lognormal	μ, σ	Mediana ($\tilde{\mu}$)

Quadro 4.4: Parâmetros de vida característica
Fonte: Nelson (2004)

Ressalta-se que em alguns casos, é possível se ter mais de um parâmetro da distribuição dependente da variável de estresse, situação esta discutida tanto em Nelson (2004, p. 105) quanto em Meeker e Escobar (1998, p. 439). Como já citado, neste trabalho assume-se que apenas um dos parâmetros da distribuição depende da variável de estresse.

4.2.5.3. Projeção da distribuição de vida para o nível de uso

Uma aplicação dos conceitos anteriores será apresentada nesta seção. Para isso, considera-se que os dados obtidos de um ensaio acelerado hipotético sejam oriundos de uma distribuição Lognormal para todos os níveis. Também, o valor de $\hat{\sigma}$ é o mesmo para todos os níveis de estresse. A segunda suposição nem sempre é satisfeita, contudo, existem transformações (NETER *et al*., 1996, p. 125) que podem ser usadas para aproximar ou até mesmo atender este requisito. Uma abordagem prática é tratar os desvios dentro do mesmo intervalo de confiança (IC) como sendo a média deste intervalo.

A seguir, o modelo de relacionamento IPL será usado em conjunto com a distribuição Lognormal, para estabelecer um modelo de aceleração que será usado para obter as

estimativas de interesse no nível de uso de um SST hipotético. Destaca-se que a descrição do modelo de aceleração inicia-se pela função densidade de probabilidade (PDF), alternativamente ao modelo apresentado anteriormente que considerou a CDF. O princípio é o mesmo, ou seja, substituir o parâmetro de vida média da PDF pelo relacionamento vida-estresse, neste caso, o IPL em sua forma linearizada. A PDF da Lognormal está representada na equação 4.8 (TRIVEDI, 2001, p. 152).

$$f(t) = \frac{1}{t\sigma_{t'}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t'-\bar{t}'}{\sigma_{t'}}\right)^2}, \tag{4.8}$$

onde:

t = valores de tempo de falha;

t' = logaritmo natural de t ou ln(t);

$\bar{t}'$ = média dos logaritmos naturais dos tempos de falha;

$\sigma_{t'}$ = desvio padrão dos logaritmos naturais dos tempos de falha.

Como apresentada no quadro 4.4, a vida característica da distribuição Lognormal é o valor da sua mediana, cuja definição é dada por:

$$\tilde{t} = e^{\bar{t}'} \tag{4.9}$$

A PDF do modelo de aceleração IPL-Lognormal pode ser obtida atribuindo, inicialmente, $\tilde{t} = \tau(V)$ em (4.10). Portanto,

$$e^{\bar{t}'} = \tau(V) = \frac{1}{K \cdot V^n} \tag{4.10}$$

Deste modo,

$$\bar{t}' = -\ln(K) - n \cdot \ln(V) \tag{4.11}$$

Substituindo (4.11) em (4.8) tem-se a PDF da IPL-Lognormal que está representada pela equação 4.12.

$$f(t,V) = \frac{1}{t\sigma_{t'}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t'+\ln(k)+n\cdot\ln(V)}{\sigma_{t'}}\right)^2} \tag{4.12}$$

A partir da PDF da IPL-Lognormal, podem ser obtidas as diversas funções relacionadas às figuras de mérito (MTTF, MTBF, taxa de falha, etc.) necessárias ao estudo da confiabilidade (FREITAS; COLOSIMO, 1997). Como exemplo, as equações 4.13 e 4.14 apresentam as suas funções confiabilidade e taxa de falha, respectivamente:

$$R(t,V) = \int_{t}^{\infty} f(t,V)dt\,, \tag{4.13}$$

ou

$$R(t,V) = \int_{t}^{\infty} \frac{1}{t\sigma_{t'}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t' + \ln(K) + n\cdot\ln(V)}{\sigma_{t'}}\right)^2} dt \tag{4.14}$$

Conseqüentemente, a função taxa de falha é descrita como:

$$\lambda(t,V) = \frac{f(t,V)}{R(t,V)} = \frac{\dfrac{1}{t\sigma_{t'}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t' + \ln(K) + n\cdot\ln(V)}{\sigma_{t'}}\right)^2}}{\displaystyle\int_{t}^{\infty} \frac{1}{t\sigma_{t'}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t' + \ln(K) + n\cdot\ln(V)}{\sigma_{t'}}\right)^2} dt} \tag{4.15}$$

O próximo passo é a estimação dos parâmetros do modelo. Em Nelson (2004), nos capítulos 3, 4 e 5 tem-se uma densa explicação dos procedimentos para a análise dos dados através de métodos gráficos (ex. papéis de probabilidade), método dos mínimos quadrados (LSE) e método da máxima verossimilhança (MLE), respectivamente. Meeker e Escobar (1998) também apresentam, em seu capítulo 8, uma descrição detalhada da aplicação do método da máxima verossimilhança para estimar os parâmetros de distribuições da família localização-escala.

Dentre as várias nomenclaturas usadas para a combinação modelo de relacionamento vida-estresse e distribuição de vida, Nelson (2004), Mettas (2003) e Freitas e Colosimo (1997) concordam na utilização do termo “modelo relacionamento-distribuição” (ex. modelo IPL-Weibull). Esta terminologia foi escolhida para ser usada nas demais seções e capítulos deste trabalho.

4.2.6. Modelos de degradação acelerada

O projeto de materiais, produtos ou sistemas de alta-confiabilidade, requer que os seus componentes individuais tenham grande confiabilidade, mesmo após longos períodos de operação ininterrupta. Freqüentemente, SSTs com tais características não manifestam nenhuma falha mesmo quando submetidos a ensaios de vida acelerados (NELSON, 2004). Uma alternativa para estes casos tem sido a obtenção da distribuição dos tempos de falha a partir de testes de degradação (MEEKER; ESCOBAR, 1998). Este tipo de teste é um enfoque alternativo aos tradicionais testes de vida e ALT.

Nos testes de degradação (DT), a resposta de interesse não é o tempo de falha, mas uma medida característica da performance do SST, que pode ser mensurada durante a sua operação (OLIVEIRA; COLOSIMO, 2004). Para muitos casos, é possível representar o relacionamento do tempo de falha com a quantidade de degradação mensurada, através de modelos que representam a curva de degradação (MEEKER; ESCOBAR, 1998). A partir do conjunto de dados de degradação, os parâmetros destes modelos são estimados e, em geral, a distribuição de vida pode ser definida como função destes parâmetros. Alternativamente, a partir do modelo de degradação é possível realizar estimativas a respeito do tempo de falha, que neste caso são denominados de tempos de *pseudofalha* (MEEKER; ESCOBAR, 1998). Com base nos tempos de *pseudofalha* estima-se a distribuição de vida do SST.

Segundo Meeker e Escobar (1998), para determinados tipos de produtos/sistemas, suas taxas de degradação em condições normais de uso se manifestam de forma muito lenta, inviabilizando a observação destas medidas em tempo aceitável. Enquadram-se nestes casos sistemas ou produtos de altíssima confiabilidade, normalmente desenvolvidos para aplicações de missão crítica (ex. sistemas de satélites). Para estas situações, Meeker e Escobar (1998) destacam que uma solução é acelerar o processo de degradação, método este conhecido como ensaio de degradação acelerado (ADT).

A utilização de ADT, como no caso de DT, também exige a definição de um modelo de degradação que descreva o comportamento, no decorrer do tempo, de uma medida característica da performance ou degradação do SST. Contudo, além do modelo de degradação, tem-se que considerar o relacionamento entre a degradação e uma variável de estresse (MEEKER; ESCOBAR; LU, 1998). ADT envolve aceleração por altos níveis de estresse (*overstress*), a fim de conseguir uma degradação mais rápida da performance do produto (NELSON, 2004). As variáveis de estresse típicas em ADT são basicamente as

mesmas de ALT, sendo citadas, com maior freqüência, a temperatura, voltagem e ciclos térmicos.

Como em ALT, para alguns materiais ou produtos já existem padrões de estresse de degradação documentados. Os demais casos necessitam definir as variáveis de estresse por meio de experimentação. Produtos de software, como já salientado, encontram-se nesta segunda categoria, pois tanto ALT quanto ADT não têm sido empregados sistematicamente em estudos experimentais em engenharia de software. Também, em ADT a forma de aplicação da variável de estresse segue os mesmos enfoques definidos para ALT (ver seção 4.2.4).

4.2.6.1. Modelo geral (Caminho de Degradação)

De acordo com Nelson (2004), as seguintes suposições são adotadas pelos atuais modelos de degradação:

a) a degradação é irreversível e sempre a performance do SST é monotonicamente pior;
b) usualmente, o modelo se aplica a um único processo de degradação e, naqueles casos que se tenha processos de degradação simultâneos, cada um requer seu próprio modelo de degradação;
c) a degradação da performance antes do início do teste é negligenciável;
d) o erro na medição da performance também é considerado negligenciável.

De acordo com Meeker e Escobar (1998), o caminho de degradação (*degradation path*) de uma unidade[24] no decorrer do tempo é representado por $D(t)$, $t > 0$. Uma amostra extraída aleatoriamente de n unidades de teste é observada em tempos predeterminados t_1, t_2, …, t_s. Para cada observação no tempo, uma medida de performance é registrada para cada unidade e referenciada como γ. Estes tempos de inspeção não precisam ser os mesmos para todas as unidades, nem tão pouco eqüidistantes. Considera-se t_{ij} como o *j-ésimo* tempo de medição da *i-ésima* unidade. A medida de degradação observada na unidade i no tempo t_{ij} é representada por γ_{ij}, e ao final do teste o caminho de degradação é registrado como pares (t_{i1}, γ_{i1}), (t_{i2}, γ_{i2}),…, $(t_{im_i}, \gamma_{im_i})$, para i=1, 2, …, n. A degradação observada γ_{ij}, da unidade i no tempo t_{ij}, é composta pela degradação atual da unidade mais o modelo de erro de medição, tal como descrita pela equação 4.16.

[24] Um espécime pertencente à amostra do material ou produto sendo testado.

$$\gamma_{ij} = D_{ij} + \varepsilon_{ij}, \quad i = 1, 2, \ldots, n \text{ and } j = 1, \ldots, m_i, \tag{4.16}$$

onde:

$D_{ij} = D(t_{ij}, \beta_{i1}, \ldots, \beta_{ik})$ é o atual caminho da unidade i no tempo t_{ij};

$\beta = (\beta_{i1}, \ldots, \beta_{ik})'$ é o vetor de coeficientes do modelo, o qual tem dimensão k;

$\varepsilon_{ij} \sim N(0, \sigma_\varepsilon)$ é o desvio residual da *i-ésima* unidade no tempo t_j.

A forma determinística de D_{ij} é normalmente baseada em análises empíricas do processo de degradação sob investigação. O vetor β corresponde à k efeitos desconhecidos, que determinam o caminho de degradação da unidade i nas medidas t_{ij}. Tipicamente, um modelo de caminho de degradação terá $k = 1, 2, 3$ ou 4 coeficientes (MEEKER; ESCOBAR, 1998). Uma fração destes coeficientes pode ser variável de unidade para unidade, e a outra parte pode ser modelada como constantes entre todas as unidades.

Em geral, é razoável assumir que os efeitos aleatórios de β são independentes dos desvios ε_{ij}. Também, assume-se que os desvios ε_{ij} são independentes e identicamente distribuídos para $i = 1, \ldots, n$ e $j = 1, \ldots, m_i$. Em função dos valores γ_{ij} serem tomados serialmente por unidade, existe um potencial para a ocorrência de autocorrelação entre os valores ε_{ij}, $j = 1, \ldots, m_i$, especialmente se os intervalos entre as observações forem muito próximos. De acordo com Meeker e Escobar (1998), em muitas situações práticas envolvendo inferência sobre a degradação de unidades de uma população ou processo, se o ajuste do modelo se mostrar adequado e se os processos de teste e medição estiverem sob controle, então a autocorrelação é tipicamente fraca e, além disso, dominada pela variabilidade existente de unidade para unidade nos valores de $\beta_1, \ldots, \beta_k$, podendo, nestes casos, ser ignorada.

No modelo geral de caminho de degradação, a proporção de falhas no tempo t é equivalente à proporção de caminhos de degradação que cruzam o nível crítico (*threshold*) D_f no tempo t. Deste modo, é possível definir a distribuição de tempos até a falha T a partir da equação 4.17, como segue:

$$F(t) = Pr(T \leq t) = Pr[D(t, \beta_1, \ldots, \beta_k) \geq D_f] \tag{4.17}$$

Para um valor fixo de D_f, a distribuição de T depende da distribuição de $\beta_1, \ldots, \beta_k$. A $F(t)$ pode ser descrita analiticamente para aqueles casos onde os modelos de degradação são mais simples. Para modelos complexos, especialmente quando D_{ij} é não-linear e mais de um

dos seus coeficientes são aleatórios, exige-se a utilização de métodos numéricos (ex. simulação de Monte Carlo) para avaliar $F(t)$.

A figura 4.5 apresenta um exemplo com caminhos de degradação construídos a partir de um conjunto de dados de degradação acelerada. Estes dados são de um experimento, descrito em Meeker e Escobar (1998, p. 574), realizado para testar a resistência de uma liga de metal. O teste foi executado aplicando-se três pesos diferentes (10, 50 e 100 gramas) e o D_f foi definido como 50 mícrons[25]. O estresse (peso) foi aplicado de forma constante para todos os três níveis. Para cada unidade alocada em um dos níveis de estresse foi construído um caminho de degradação.

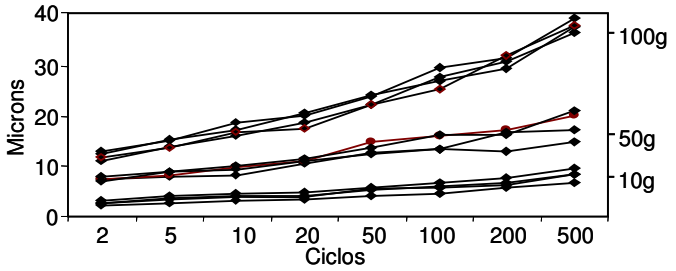

Figura 4.5: Exemplo de caminhos de degradação
Fonte: Meeker e Escobar (1998)

4.2.6.2. Obtenção da distribuição de vida a partir de modelos de degradação acelerada

No exemplo da figura 4.5, nenhum dos três níveis de estresse foi suficiente, no período de 500 ciclos, para levar os caminhos de degradação até o valor do limiar de falha estipulado em $D_f = 50$ mícrons. Assumindo que os caminhos de degradação tivessem atingido o valor de D_f, os tempos t_{im_i} seriam utilizados para a obtenção da distribuição dos tempos de falha em cada nível de estresse. A partir destes dados, os procedimentos de análise de dados utilizados em ALT, descritos na seção 4.2.5.2, seriam utilizados para estimar a distribuição de vida na condição padrão de uso.

No caso do caminho de degradação não atingir o valor de D_f, é necessário a adoção de um método para estimar os tempos de *pseudofalha* a partir do conjunto de dados de degradação, a fim de posteriormente encontrar a distribuição de vida para cada nível de estresse. Dentre os métodos descritos em Meeker e Escobar (1998), Oliveira e Colosimo (2004) e Nelson (2004), a seguir será apresentado o método de aproximação ou também

[25] 50 *mícrons* correspondem a 0,005 centímetro.

denominado de análise de degradação acelerada aproximada. Este método foi escolhido, haja vista sua facilidade de implementação. A seguir uma descrição das principais etapas deste método, cujo objetivo é a estimação da distribuição de vida ($F(t)$) a partir do conjunto de dados de degradação acelerada:

a) Para a unidade i o modelo $\gamma = D_{ij} + \varepsilon$ é ajustado. Os efeitos do modelo são considerados fixos para cada unidade e aleatórios entre elas;
b) As estimativas do vetor $\beta = (\beta_{i1},\ldots, \beta_{ik})'$ para a unidade i ($\hat{\boldsymbol{\beta}}_i$) podem ser obtidas através dos métodos LSE ou MLE;
c) Resolve-se a equação $D(t, \hat{\boldsymbol{\beta}}_i) = \mathrm{D_f}$ para t e a solução é chamada de $\hat{t}_i$;
d) Repetir os procedimentos anteriores para cada caminho de degradação a fim de obter os tempos de *pseudofalha* $\hat{t}_1, \ldots, \hat{t}_n$;
e) Posteriormente, aplicam-se os métodos para análise de dados de ensaios de vida acelerados, descritos na seção 4.2.5.2, para se estimar a $F(t)$ em cada nível de estresse e para o nível de uso.

Como resultado dos passos anteriores, tem-se a $\hat{F}(t)$ para o nível de estresse na condição padrão de operação, o que permite a análise quantitativa da confiabilidade do material, produto ou sistema em suas condições de projeto.

4.3. PLANEJAMENTO DO ENSAIO ACELERADO

Como verificado nas seções anteriores, a realização de ensaios acelerados exige a especificação de uma série de elementos, cuja definição adequada é determinante para o sucesso do experimento. Muitas vezes, existem restrições de custo, tempo e dos próprios mecanismos de aceleração que devem ser administradas de forma a se maximizar os resultados obtidos com o ensaio. Antes de iniciar a execução dos testes, faz-se necessário planejá-los de forma a se minimizar possíveis desvios das estimativas (ex. vida média, taxa de falhas, etc.) que serão obtidas a partir dos dados produzidos nos níveis elevados de estresse.

Deste modo, esta seção apresenta uma visão geral dos principais elementos a serem contemplados no planejamento de um ensaio acelerado, os quais são baseados no roteiro proposto em Freitas e Colosimo (1997, p. 231). Neste roteiro existem elementos que já foram discutidos em maior detalhamento nas seções anteriores deste capítulo, e neste caso serão

abordados resumidamente. O roteiro adotado divide os principais elementos envolvidos no planejamento de um ensaio acelerado em dois grupos:

a) Forma do teste:
   - Escolha da medida de performance;
   - Condições de teste;
   - Variável de estresse;
   - Forma de aplicação do estresse;
   - Mecanismo de censura.

b) Plano experimental:
   - Número de níveis da variável de estresse;
   - Determinação dos níveis da variável de estresse;
   - Proporção de alocação em cada nível;
   - Tamanho da amostra.

O primeiro grupo é composto por elementos que geralmente são determinados a partir das condições ambientais da empresa ou local de realização dos testes (FREITAS; COLOSIMO, 1997). Já o segundo, contém elementos determinados de forma quantitativa e será objeto de um maior detalhamento.

4.3.1. Forma do teste

A seguir tem-se uma descrição sucinta dos elementos que constituem a forma do teste:

- Escolha da medida de performance: Tanto para ALT quanto ADT existem normas técnicas que definem quais são as medidas de performance a serem utilizadas e como medi-las. A medida de performance deve ser monitorada de forma a se avaliar a condição de falha do sistema ou a degradação de sua performance.
- Determinação das condições de teste: As condições reais de operação do SST devem ser preservadas durante os testes, exceto com relação aos níveis da variável de estresse. Para isso, é fundamental se realizar a prévia caracterização da operação do SST, em condições reais de uso, a fim de se buscar criar as mesmas condições em nível experimental.
- Definição das variáveis de estresse: Assim como no primeiro item, para muitos produtos existem padrões (normas) para testes envolvendo a aceleração da vida/degradação de um SST através da elevação dos níveis de algum tipo de variável

de estresse. Nos demais casos, abordagens experimentais são adotadas objetivando verificar qual estresse, de fato, acelera a vida/degradação do produto.

- Definição da forma de aplicação da carga de estresse: A escolha da forma da aplicação da carga de estresse (ver seção 4.2.4) depende de como o sistema é sobrecarregado durante sua operação em campo, bem como de restrições práticas e teóricas. A caracterização da operação do sistema, citada no segundo item, é de grande ajuda para esta etapa.
- Mecanismo de censura: No caso dos testes do tipo ADT, faz-se necessário definir o limiar de degradação (D_f) que, ao ser atingido ou ultrapassado pela medida de performance, se declara que ocorreu uma falha do SST. Nos testes do tipo ALT, a censura pode ser por falha (tipo II) ou por tempo (tipo I). A censura por falha é usada, geralmente, quando se tem pouca ou nenhuma informação sobre o SST. Desta forma, garante-se um número mínimo de falhas para realizar a análise estatística dos dados. Em contrapartida, neste tipo de censura não é possível determinar uma duração exata do teste, haja vista que o mesmo ocorre até a falha do sistema. No caso da censura por tempo, normalmente sua aplicação ocorre quando existem informações históricas do SST, permitindo assim definir a duração do teste (D_t). Segundo Freitas e Colosimo (1997, p. 232), a censura por tempo é a mais utilizada, haja vista a necessidade de definição prévia da duração dos testes. Esta necessidade normalmente origina-se de restrições de custo e tempo de conclusão dos projetos.

4.3.2. Plano experimental

A seguir uma descrição de cada elemento que compõe o plano experimental:

- Número de níveis de estresse: O número de níveis deve ser balanceado a partir dos objetivos e das restrições do teste. Mettas (2003) lembra que são exigidos pelo menos dois níveis de aceleração para a implementação de ALT, o que também vale para ADT. Não existe um limite máximo para o número de níveis de estresse, contudo, em termos práticos um número elevado apresenta diversas restrições com relação ao tempo disponível para a realização dos testes, dentre outros aspectos (FREITAS; COLOSIMO, p. 233). Na revisão da literatura a maioria dos planos citados apresentou três ou quatro níveis.

- Níveis de estresse: Os níveis de estresse não devem ultrapassar os limites de projeto do SST. Normalmente, estes níveis estão fora dos limites de especificação e dentro dos limites de projeto (METTAS, 2003). Ultrapassando os limites de projeto, muitos sistemas apresentarão modos de falha indesejáveis ao estudo. A figura 4.6 ilustra estes conceitos.

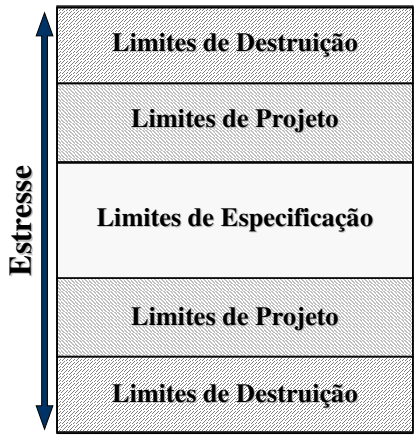

Figura 4.6: Faixas típicas de estresse
Fonte: Mettas (2003)

- Proporção de alocação: A proporção de alocação define a distribuição do número total de testes entre os níveis de estresse. Por exemplo, um estudo com três níveis de estresse e um total de *n* testes, onde cada nível de estresse recebe a mesma quantidade de ensaios, tem uma proporção de alocação de 1/3, ou seja, *n*/3 ensaios em cada nível de estresse.
- Tamanho da amostra: A quantidade (*n*) de unidades que serão testadas. Normalmente, uma unidade é testada apenas no nível de estresse para o qual a unidade foi alocada. Nelson (2004), Freitas e Colosimo (1997) e Meeker e Escobar (1998) apresentam procedimentos para calcular o valor de *n*. O apêndice A descreve o procedimento que será usado pelo método proposto (ver capítulo 5), o qual está definido em Nelson (2004, p. 327).

A definição dos valores dos quatro elementos descritos anteriormente ocorre de acordo com a proposta adotada para o plano experimental. Na literatura revisada (FREITAS; COLOSIMO, 1997; MEEKER; ESCOBAR, 19998; NELSON, 2004), três propostas são apresentadas como sendo as mais utilizadas, sendo elas: planos tradicionais, planos ótimos e planos de compromisso.

Cada proposta possui mais de uma configuração, compondo um rol de opções que podem ser aplicadas durante o planejamento. Além disso, ressalta-se que a acuracidade dos resultados obtidos com estas propostas depende dos métodos de estimação de parâmetros e dos modelos teóricos adotados. Estes modelos são as distribuições de vida, ajustadas em cada nível de estresse, juntamente com o modelo de relacionamento vida-estresse usado para estimar a distribuição de vida na condição de interesse. Neste sentido, verificou-se que a literatura revisada enfatiza aqueles planos que: o relacionamento vida-estresse seja linear (ex. IPL-Weibull), os métodos de estimação usados sejam tanto LSE quanto MLE e as distribuições de vida sejam da família (log-)localização-escala.

Uma apresentação detalhada das três propostas de planejamento foge do escopo deste trabalho. Maiores detalhes sobre o assunto podem ser encontrados no capítulo 20 de Meeker e Escobar (1998), capítulo 6 de Nelson (2004) e capítulo 7 de Freitas e Colosimo (1997). Portanto, a seguir será apresentada uma breve explanação sobre cada proposta, a fim de introduzir aqueles conceitos que serão utilizados durante a definição do planejamento para o método proposto.

4.3.2.1. Planos tradicionais

Os planos tradicionais consistem em fixar três ou quatro níveis de estresse igualmente espaçados e alocar o mesmo número de testes em cada nível. Segundo Freitas e Colosimo (1997, p. 235), este tipo de plano é muito utilizado na prática, porém oferece estimadores menos precisos quando comparados aos planos ótimos e de compromisso, por considerar o mesmo número de testes em cada nível de estresse. Os resultados menos precisos ocorrem nos níveis baixos de estresse, devido ao menor número de falhas que normalmente ocorrem nestes níveis.

Em Meeker e Escobar (1998), tem-se um exemplo de aplicação de um plano tradicional para o ensaio acelerado de uma cola adesiva em três níveis de temperatura. A tabela 4.1 apresenta este plano. Como se pode verificar, o nível mais baixo (110ºC) é mais que o dobro da temperatura na condição de uso (50ºC). Por se tratar de um plano tradicional, os níveis de estresse são espaçados igualmente, da mesma forma como é definida a proporção de alocação. Neste caso, o tamanho da amostra (300) foi definido mediante a disponibilidade de recursos, não tendo sido aplicado nenhum método quantitativo para a sua determinação.

Tabela 4.1: Exemplo de plano tradicional com três níveis de estresse
Fonte: Adaptado de Meeker e Escobar (1998, p. 537)

Nível da Temp. (ºC)	Alocação	
	Proporção π_i	Número de testes n_i
50		
110	1/3	100
130	1/3	100
150	1/3	100

4.3.2.2. Planos ótimos

De acordo com Nelson (2004, p. 326), este é o tipo de plano que oferece os estimadores mais precisos para as condições de projeto. São definidos apenas dois níveis de estresse, nível alto (N_a) e nível baixo (N_b). Meeker e Escobar (1998, p. 542) salientam que o valor de N_a é definido em termos do máximo permitido, e os valores de N_b e da sua proporção de alocação (π_b) devem ser definidos de forma a minimizar a variância dos estimadores de interesse.

Diferente dos planos tradicionais a alocação dos planos ótimos é desigual. Segundo Nelson (2004, p. 321), esta alocação é baseada na fração p que minimiza a variância do estimador da medida de interesse no nível de uso da variável de estresse. Assumindo um tamanho de amostra n, o número de testes alocados no N_b é o inteiro mais próximo de np, sendo que os demais testes são alocados no N_a.

Freitas e Colosimo (1997, p. 235) salientam que "[...] os planos ótimos têm o inconveniente de serem utilizados com somente dois níveis de estresse, dificultando a extrapolação dos resultados para as condições de uso". Esta afirmação corrobora com vários argumentos apresentados em Nelson (2004, p. 326), que abordam os aspectos desfavoráveis dos planos ótimos. A tabela 4.2 apresenta um exemplo de plano ótimo, definido a partir dos mesmos dados do ensaio acelerado da cola adesiva citado na seção anterior.

Tabela 4.2: Exemplo de plano ótimo
Fonte: Adaptado de Meeker e Escobar (1998, p. 543)

Nível de estresse		Alocação	
Condição	Temperatura (ºC)	Proporção π_i	Número de testes n_i
Uso	50		
N_b	95	.71	212
N_a	120	.29	88

4.3.2.3. Planos de compromisso

Estes planos adotam de três a quatro níveis de estresse, sendo que a proporção de alocação é desigual como ocorre nos planos ótimos. Um exemplo de plano de compromisso são os chamados planos de *Meeker-Hahn* (NELSON, 2004, p. 343). Nestes, são adotados três níveis de estresse com a alocação N_b = 4/7, N_i = 2/7, e N_a = 1/7. Como exemplo desta distribuição 4:2:1, para uma amostra de n testes tem-se $N_b = 4n/7$, $N_i = 2n/7$, e $N_a = n/7$.

De forma geral, os planos de compromisso, assim como nos planos ótimos, assumem que o nível alto seja escolhido por considerações práticas (FREITAS; COLOSIMO, 1997, p. 235). O nível intermediário é o ponto médio entre o nível baixo e o nível alto, ou seja, $N_i = (N_a + N_b) / 2$, garantindo desta forma que os níveis sejam eqüidistantes. Portanto, resta a definição do nível baixo que, para Nelson (2004, p. 326), deve ser feita considerando a acuracidade das estimativas que se pretende obter no nível de uso. Desta forma, busca-se sempre que possível um plano que minimize a variância destas estimativas. Para os planos de *Meeker-Hahn*, Freitas e Colosimo (1997, p. 237) apresentam uma abordagem para a seleção dos níveis baixo e intermediário com base em tabelas padronizadas. Em outros exemplos de planos de compromisso, como em Nelson (2004, p. 326), estes níveis são definidos com base em restrições do projeto. A tabela 4.3 apresenta um exemplo de plano de compromisso. Neste, adotou-se uma política de alocação onde o nível intermediário recebe a menor parcela de testes, sendo a maior atribuída ao nível baixo, o que corresponde à quase o dobro do nível alto.

Tabela 4.3: Exemplo de plano de compromisso
Fonte: Adaptado de Meeker e Escobar (1998, p. 546)

Nível de estresse		Alocação	
Condição	Temperatura (ºC)	Proporção π_i	Número de testes n_i
Uso	50		
N_b	78	.52	156
N_i	98	.20	60
N_a	120	.28	84

Um plano de compromisso similar ao da tabela 4.3 é apresentado em Nelson (2004, p. 326), onde foram comparados três planos (tradicional, ótimo e de compromisso), verificando que o plano de compromisso proporcionou uma acuracidade das estimativas que se situou entre os resultados obtidos com os planos tradicional e ótimo. Mais comparações a respeito das três abordagens são descritas em Nelson (2004, p. 324) para dados completos, e em

Nelson (2004, p. 341) para testes com censura do tipo I. A próxima seção apresenta uma síntese de trabalhos que aplicam ou discutem a utilização de ensaios acelerados em software.

4.4. ENSAIOS ACELERADOS APLICADOS EM PRODUTOS DE SOFTWARE

A revisão da literatura considerou trabalhos de pesquisa envolvendo ensaios acelerados do tipo quantitativos (ALT e ADT) aplicados a sistemas ou componentes de software. Após consulta em diversos periódicos, bibliotecas *on-line* e sistemas de busca de publicações, nas áreas de engenharia de software experimental, *dependabilidade* computacional, engenharia de confiabilidade de software, dentre outras correlatas, foram encontrados dois trabalhos (EHRLICH *et al.*, 1998; CHAN, 2004) que abordam a utilização de ALT em estudos envolvendo produtos de software. Destes dois trabalhos, apenas Ehrlich *et al.* (1998) realiza a aplicação deste método. Juntamente com estes dois trabalhos, um terceiro (CHILLAREGE; GOSWANI; DEVARAKONDA, 2002) será apresentado, o qual propõe e valida experimentalmente um modelo de aceleração de falhas para software baseado na aceleração por taxa de uso.

Em Ehrlich *et al.* (1998), foi utilizado o método ALT como parte de um estudo de análise de confiabilidade do software de um sistema de recuperação de redes de telecomunicações. O sistema analisado foi projetado para a geração de alarmes e recuperação da rede, mediante a ocorrência de falhas causadas pelo rompimento de cabos de fibra óptica. Segundo os autores, estes são eventos altamente críticos e ocorrem com pouquíssima freqüência. Os autores citam que este tipo de evento ocorre em torno de 10 vezes por ano na rede da AT&T nos Estados Unidos, empresa onde foi realizada a pesquisa. Ressalta-se que em Hecht (1993), eventos raros foram identificados como sendo uma das principais causas de falhas de software, exatamente pelo fato da dificuldade em se reproduzir suas condições de ativação durante a operação normal do sistema (ver seção 2.3).

Ehrlich *et al.* (1998) justificam o uso de ALT por causa da baixa freqüência de ocorrência dos eventos de interesse, os quais dão início à execução das rotinas de recuperação, o que é necessário para a condução dos testes e posterior análise de sua confiabilidade. Segundo os autores, testar o sistema sob condições normais de operação pode ser caro e consumir muito tempo, sendo até inviável em certas condições.

Dentre as abordagens para a realização de ALT, Ehrlich *et al.* (1998) adotaram a aceleração através da elevação na taxa de uso (ver seção 4.2.3.1). Como variável de aceleração, foi definida a carga de processamento em segundo plano (*background*

processing), pois os autores verificaram que a interação entre o processamento em segundo plano e as transações relacionadas aos eventos de falha criava condições para a ocorrência de falhas no sistema de recuperação. Como a taxa de processamento em segundo plano era baixa na condição normal de operação, a probabilidade de ocorrência destas interações também era pequena, principalmente se levando em conta a reduzida probabilidade de manifestação dos eventos de falha. Neste sentido, os autores adotaram três níveis de aceleração para a taxa de processamento em segundo plano, equivalendo a 10, 100 e 200 vezes a taxa atual (condições de campo). A ordem de execução dos testes seguiu um projeto de experimentos baseado em dois quadrados Latinos 4 x 4, resultando em 32 ensaios. Esta configuração foi justificada pelos autores, para considerar a possibilidade dos efeitos de um tratamento se sobreporem aos efeitos dos demais tratamentos, já que devido à limitação de recursos para a pesquisa, todos os ensaios foram executados seqüencialmente no mesmo equipamento. Para cada um dos três níveis de estresse, foram executados os 32 ensaios com duração média de 30 horas, resultando em um tempo total de experimentação de aproximadamente 90 horas e 30 minutos, considerando um intervalo de repouso de 15 minutos entre cada lote de 32 ensaios. Este intervalo de repouso foi necessário para o sistema restabelecer suas configurações iniciais antes de entrar no próximo nível de taxa de uso. As falhas foram analisadas utilizando um modelo de regressão de Poisson. A partir do modelo de regressão, foi possível estimar o tempo médio de falha para a condição de uso, ou seja, com o nível de processamento em segundo plano na condição normal de operação. Uma análise complementar considerou a utilização da distribuição de Weibull para modelar os tempos de falha, resultando em valores próximos aos obtidos com o modelo de regressão de Poisson.

Em Chan (2004), discute-se a importância de uma abordagem de aceleração de falha tanto para hardware quanto para software, de forma conjunta, já que atualmente ambas tecnologias estão integradas de forma predominante em diversos tipos de sistemas. O autor salienta que apesar deste cenário, as atuais abordagens de ensaios acelerados têm considerado apenas a aceleração de falhas de hardware. O tipo de ensaio acelerado abordado por Chan (2004) é o qualitativo, o qual foi chamado em seu trabalho de teste de estresse acelerado (*accelerated stress testing - AST*). O método de ALT foi citado e discutido superficialmente no final do trabalho, onde se fez uma comparação entre ALT, AST e outras abordagens qualitativas. Nesta comparação, o autor comenta que em AST o nível de aceleração das variáveis de estresse não exige o mesmo compromisso que nos testes baseados em ALT, já que em AST não se pretende inferir medidas de confiabilidade do sistema em seu nível de operação usual.

A abordagem qualitativa dos testes do tipo AST, descritos em Chan (2004), se relaciona com o presente trabalho em 2 aspectos. O primeiro ao destacar a importância da aceleração de falhas também em software, e não apenas em hardware, salientando a importância de métodos sistematizados que reduzam tempo e custo, auxiliando na melhoria da confiabilidade dos componentes de software que estão integrados ao hardware. O segundo ponto tem relação com o conceito de estresse de aceleração em software. Em hardware, existem diversos padrões de variáveis de estresse definidos para determinados tipos de produtos e materiais, contudo em software não existem tais especificações. Neste aspecto o artigo comenta, mesmo que superficialmente, o conceito de variável de estresse como sendo aquela condição de ativação das falhas de software. Uma crítica ao estudo apresentado em Chan (2004), se refere à sua descrição dos ensaios do tipo ALT. A descrição apresentada não explorou os aspectos essenciais para o entendimento desta técnica. Além disso, o artigo falha em atribuir ao método ALT uma suposta limitação de se utilizar apenas níveis de estresse similares aos estresses de campo, sendo este compromisso uma premissa necessária para a posterior extrapolação da distribuição de vida para o nível de uso. De fato, a liberdade para adotar níveis elevados de estresse, mais ou menos diferentes daqueles experimentados em campo, também existe em ALT, desde que o modelo usado para relacionar estes níveis de aceleração, com o nível de uso, seja adequado.

Com um enfoque diferente dos dois trabalhos descritos anteriormente, Chillarege, Goswani e Devarakonda (2002) examinam e apresentam evidências empíricas de uma teoria proposta em Devarakonda (1990 apud CHILLAREGE; GOSWANI; DEVARAKONDA, 2002) para a aceleração de falhas em software. Esta teoria não tem relação direta com as técnicas ALT/ADT, contudo a mesma possui alguns conceitos que se relacionam com o método que será apresentado no capítulo 5.

A teoria de aceleração discutida em Chillarege, Goswani e Devarakonda (2002) considera que a aceleração da falha é conseguida minimizando a latência da falta e do erro, e também aumentando a probabilidade de uma falta causar uma falha. Neste sentido, a teoria assume que a aceleração de falha é obtida a partir das seguintes condições:

Latência da Falta $\rightarrow 0$;

Latência do Erro $\rightarrow 0$;

Pr(Falta) $\rightarrow 1$, onde *Pr(Falta)* é a probabilidade de uma falta causar uma falha.

Os autores sugerem dois controles básicos para implementar estas condições, os quais são: o tamanho da falta e a carga de trabalho. O primeiro controle se refere à injeção de uma

falta (JALOTE, 1994), cuja amplitude dos seus efeitos incrementa a probabilidade de ocorrer uma falha no software. No segundo, o controle da carga de trabalho permite decrementar a latência do erro a partir do incremento do uso do sistema, enfoque este também adotado em Ehrlich *et al*. (1998) ao selecionar valores elevados para a taxa de processamento em segundo plano (ver seção 4.4).

A validação da teoria descrita em Chillarege, Goswani e Devarakonda (2002) foi baseada em experimentação e objetivou a obtenção dos tempos de falha de um servidor NFS (*Network File System*). Foram adotados dois níveis de carga (baixo=15%-30% e alto=25%-50%) para a utilização da CPU do servidor. A carga de trabalho foi baseada em um conjunto de operações de acesso ao sistema de arquivos do servidor NFS, realizadas por um sistema cliente, as quais possuíam faltas injetadas com o objetivo de provocar erros no processamento do servidor. De acordo com Chillarege, Goswani e Devarakonda (2002), a diferença na fração de falhas entre os níveis de aceleração (baixo e alto) foi substancial, tendo resultado em um aumento de 53% para 65% na probabilidade de falha de um nível para o outro. Também, os autores destacam que houve um decremento na propagação do erro como conseqüência da aceleração de falha, o que faz sentido, pois o incremento na aceleração deveria decrementar a latência do erro e conseqüentemente reduzir a chance para o erro se propagar.

4.5. CONSIDERAÇÕES FINAIS

A técnica de ensaio de vida acelerado tem sido cada vez mais utilizada em várias áreas da indústria, em resposta à crescente exigência de redução de custos e rapidez na obtenção de dados de vida de produtos. Contudo, como salientado em Chan (2004), a sua aplicação tem sido restrita apenas aos testes de hardware, mesmo quando os sistemas são compostos de hardware e software de forma integrada (ex. sistemas embarcados). Uma suposição que aqui se apresenta, para explicar este cenário, diz respeito aos métodos de aceleração, especialmente quando se trata de aceleração por altos níveis de estresse (ver seção 4.2.3.2). Estes métodos são normalmente baseados em leis que regem a física das falhas de materiais e produtos.

Em software, o conceito de física da falha não é bem estabelecido tal como nas demais áreas. Isto se deve à própria natureza das falhas de software, que não podem ser explicadas por leis ou fenômenos físicos/químicos. A diferença entre a "física" das falhas de software, em relação às demais áreas, motivou o capítulo 2, o qual teve por objetivo descrever como ocorre esse processo no âmbito do software.

Outro aspecto que dificulta a aplicação de ensaios acelerados em software é com relação ao conceito de desgaste/envelhecimento. Nas demais áreas, o envelhecimento também está diretamente associado às propriedades físico-químicas dos materiais que compõem os produtos sob teste. Em software, não é possível se aplicar leis tais como de Arrhenius ou Eyring (ver quadro 4.3). De fato, a fenomenologia do envelhecimento de software é bem diferente do que ocorre no mundo físico, o que dificulta sua compreensão e conseqüentemente a utilização das técnicas de ensaios acelerados. A necessidade por compreensão deste fenômeno, a fim de utilizar os ensaios acelerados para a sua análise, motivou a apresentação do capítulo 3, com objetivo de esclarecer os principais aspectos que compõem o processo de envelhecimento em produtos de software.

Neste trabalho, busca-se com a compreensão do processo que leva às falhas causadas pelo envelhecimento de software, estabelecer os conceitos necessários à teoria de ensaios acelerados, de forma que se possa aplicá-la na análise de dados de falhas por envelhecimento de software. As principais diferenças deste trabalho em relação à Ehrlich *et al*.(1998) e Chillarege, Goswani e Devarakonda (2002), diz respeito ao método de aceleração e a aplicação voltada ao envelhecimento de software. Nos dois artigos citados, o problema do envelhecimento de software não é abordado. Além disso, o método adotado em ambos os trabalhos foi implementar a aceleração das falhas por meio da elevação da taxa de uso controlando a carga de trabalho. O método proposto neste trabalho também utiliza o controle da carga de trabalho, mas para ajustar os níveis de estresses utilizados para acelerar os efeitos do envelhecimento de software. Esta abordagem será apresentada no próximo capítulo.

CAPÍTULO 5

MÉTODO DE ACELERAÇÃO DO ENVELHECIMENTO DE SOFTWARE BASEADO EM ENSAIOS DE VIDA ACELERADOS QUANTITATIVOS

5.1. INTRODUÇÃO

O capítulo 4 descreveu duas técnicas de ensaios acelerados quantitativos (ALT e ADT). A primeira é a abordagem tradicional, em que os níveis de aceleração são suficientes para causar a manifestação das falhas em tempo aceitável. A segunda, que em muitos casos complementa a primeira, é aplicada naqueles estudos em que mesmo adotando níveis elevados de estresse ou de taxa de uso, o tempo para a obtenção das falhas se demonstra muito elevado.

Como já discutido no capítulo 3, muitos experimentos envolvendo envelhecimento de software exigem um tempo elevado para a manifestação de falhas. Devido a estas características, o método proposto neste capítulo considera a utilização de testes de degradação acelerados (ADT) como a técnica que será aplicada durante os ensaios de aceleração das falhas por envelhecimento de software. ADT tem a vantagem de ser suficientemente flexível para tratar tanto aqueles casos em que as falhas ocorrem dentro do período de testes, como também as situações onde a duração dos testes não é suficiente para se observar falhas no sistema sob teste (SST). Neste último cenário, ADT compensa a ausência de falhas analisando os dados de degradação do SST. ADT é uma abordagem baseada na aceleração da degradação através da adoção de variáveis de estresse em níveis mais elevados do que o usual. Deste modo, o método proposto não contempla a aceleração de falhas através da elevação da taxa de uso, dando ênfase para a aceleração por meio de variáveis de estresse. Apesar de não ser apresentado, a adoção de aceleração por taxa de uso não implica em modificações substanciais no método, sendo facilmente implementado com base nos procedimentos que serão apresentados nas próximas seções deste capítulo.

O método proposto está organizado em 4 processos principais, os quais estão representados na figura 5.1. No primeiro processo, é realizada a seleção do fator de envelhecimento, o qual corresponde ao conceito de variável de estresse (ou estresse de aceleração). O fator de envelhecimento é um componente fundamental desta proposta, haja

vista que os demais processos dependem da sua definição. O segundo processo refere-se ao planejamento e execução do ADT, com o objetivo de acelerar os efeitos do envelhecimento de software para obter os tempos de falha ou *pseudofalha* do SST. O terceiro processo trata da modelagem do relacionamento vida-estresse, chamado neste método de estresse-envelhecimento acelerado. No quarto processo é realizada a estimação da distribuição de vida para o nível de uso do sistema, que é o principal objetivo desta proposta.

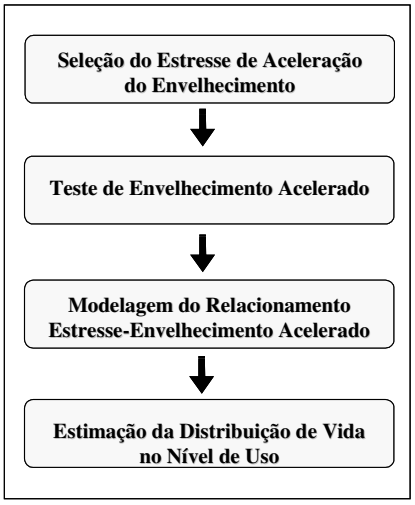

Figura 5.1: Principais processos do método proposto

Com exceção do primeiro processo, os demais conservam os procedimentos utilizados em ensaios de degradação acelerados aplicados nas demais áreas da indústria. Este é um aspecto positivo do método proposto, o qual reutiliza o corpo de conhecimento estabelecido para outras aplicações e que são aderentes aos estudos voltados para o envelhecimento de software. Em contrapartida, os resultados experimentais obtidos neste trabalho (ver capítulo 6) servirão como prova de conceito da viabilidade de se aplicar estes procedimentos a estudos envolvendo aceleração de falhas de software, em especial de envelhecimento de software.

Cada um dos quatro processos será descrito em termos de suas entradas, ferramentas/técnicas utilizadas e saídas. Esta estrutura está representada na figura 5.2 e se baseia na mesma representação de processos adotada pelo guia PMBOK (PMI, 2004). Esta forma de representação foi escolhida pela sua simplicidade. Também, por ser uma representação de processos conhecida, como é o caso do guia PMBOK, a sua integração com outros processos organizacionais voltados para o desenvolvimento de produtos, gerenciamento da qualidade (ex. confiabilidade, disponibilidade, etc.), dentre outros, se torna de fácil implementação.

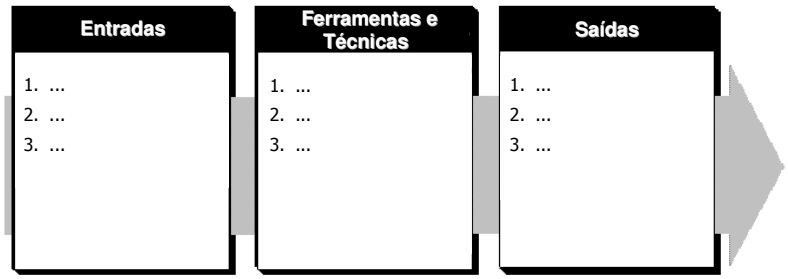

Figura 5.2: Estrutura de representação dos processos

Antes de iniciar o detalhamento de cada processo que compõe o método proposto, uma discussão sobre os principais aspectos envolvidos na aceleração do envelhecimento de software se faz necessária.

5.2. ACELERAÇÃO DO ENVELHECIMENTO DE SOFTWARE

A diferença na aplicação das técnicas ADT/ALT em experimentos de software, em comparação com as demais áreas, reside basicamente nos mecanismos de falha/degradação e na forma de aceleração destes mecanismos. Nas outras áreas, os mecanismos de falha e os estresses utilizados para a aceleração destes mecanismos são baseados em leis bem estabelecidas.

No envelhecimento de software, os mecanismos que causam a degradação progressiva do sistema não são de natureza física. Estes podem ser compreendidos como os efeitos degenerativos causados pelas sucessivas ativações das faltas relacionadas ao envelhecimento, que ocorrem durante o período de longevidade (tempo de execução) do sistema.

A partir desta definição, o conceito de estresse de aceleração, proveniente da teoria de ensaios acelerados, deve ser redefinido a fim de ser aplicado aos estudos experimentais em envelhecimento de software. Portanto, esta pesquisa considera como estresse de aceleração do envelhecimento de software os padrões de ativação de faltas relacionadas ao envelhecimento, ou seja, faltas do tipo FRE (ver seção 3.2.2). Deste ponto em diante neste documento, estes padrões de ativação serão chamados de fator de envelhecimento (F_E). A priori, considera-se que a identificação do F_E será realizada de forma experimental, haja vista a ausência de estudos prévios que forneçam subsídios teóricos para sua definição.

A fim de explorar o conceito de F_E, será utilizado como exemplo o caso citado na seção 3.1, referente ao problema de vazamento de memória nos *switches* da linha *Catalyst*. Naquele exemplo, a ativação da falta que causava o vazamento de memória ocorria sempre que uma

autenticação de sessão TELNET falhasse ou nos casos onde a duração de uma sessão TELNET ocorresse em um período muito curto (CISCO SYSTEMS, 2000). Deste modo, consideram-se como padrões para a ativação da falta que provoca o vazamento de memória as duas condições citadas anteriormente. Portanto, o F_E pode ser representado por apenas um, ou ambos os padrões de ativação, que ao serem submetidos ao SST causam o seu envelhecimento.

Como ilustrado pelo exemplo anterior, o método proposto adota um enfoque orientado aos fatores que ativam as faltas que provocam os efeitos do envelhecimento de software, exigindo que a caracterização do envelhecimento seja realizada sob a perspectiva das suas causas e não dos efeitos. Como já salientado no capítulo 3, os trabalhos antecessores nesta área têm se dedicado, principalmente, à identificação e mensuração dos efeitos do envelhecimento, o que torna esta nova abordagem uma das principais contribuições deste trabalho. A seguir será apresentado o processo de seleção do estresse de aceleração do envelhecimento, aqui denominado de F_E.

5.3. SELEÇÃO DO FATOR DE ENVELHECIMENTO

Inicialmente, é importante ressaltar que o objetivo desta etapa não é a verificação da presença ou não do envelhecimento de software no SST. O método assume que esta verificação já foi realizada e que seu resultado tenha sido positivo. A identificação do envelhecimento pode ser realizada utilizando uma das abordagens descritas na revisão da literatura apresentada no capítulo 3.

Na seção anterior, o F_E foi definido como sendo um ou mais padrões de ativação de faltas do tipo FRE. Estes padrões estão diretamente ou indiretamente relacionados às requisições submetidas ao SST e, na maioria dos casos, podem ser capturados através dos parâmetros da carga de trabalho do sistema. A caracterização da carga de trabalho do SST não faz parte desta proposta, a qual assume que esta atividade tenha sido realizada previamente. Em Jain (1991, p. 71), são descritas diversas técnicas para a caracterização e modelagem da carga de trabalho de sistemas computacionais, as quais podem ser empregadas para este propósito.

O processo de seleção do F_E é baseado na caracterização do envelhecimento, onde são avaliados os parâmetros da carga de trabalho, juntamente com outras variáveis relacionadas, a fim de identificar aqueles padrões que ativam as FRE. Esta caracterização está dividida em duas fases:

a) Teste experimental dos fatores candidatos: objetiva testar experimentalmente os fatores candidatos à ativação das FRE, com base nos parâmetros da carga de trabalho e demais variáveis relacionadas. O resultado desta etapa é a medição dos efeitos do envelhecimento no comportamento do sistema, frente às alterações controladas na carga de trabalho.
b) Análise dos efeitos dos fatores sobre o envelhecimento: A partir dos dados de medição, busca-se estimar analiticamente o grau de influência que os fatores candidatos exercem sobre o envelhecimento de software do sistema investigado. Com base nestes resultados cria-se um ranking dos fatores com maior contribuição sobre os efeitos do envelhecimento.

A partir do ranking gerado no passo (b) supracitado, o F_E é definido como o parâmetro, ou a combinação de parâmetros, da carga de trabalho que exerce maior influência sobre os efeitos do envelhecimento de software. Apesar dos parâmetros da carga de trabalho serem o ponto de partida para a seleção do F_E, é possível se ter outras variáveis que façam parte da composição do F_E.

5.3.1. Teste experimental dos fatores candidatos

A seleção do F_E envolve experimentação, haja vista a ausência de referencial teórico na área. Segundo Montgomery (2005, p. 12), um enfoque científico para planejar experimentos é o método de planejamento estatístico de experimentos (DOE). Este enfoque baseia-se no planejamento de experimentos de forma que os seus resultados possam ser analisados por métodos estatísticos. Dentre as aplicações do DOE, Montgomery (2005) descreve sua utilização com objetivo de caracterização de processos (*screening experiments*). Este tipo de aplicação é o objetivo desta etapa do método, voltada para a seleção do F_E. A figura 5.3 apresenta os principais elementos do DOE.

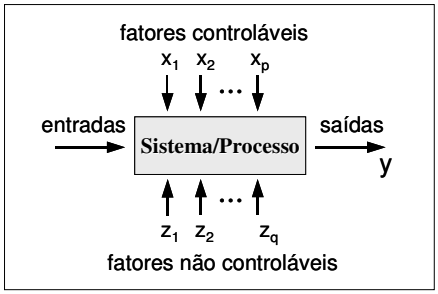

Figura 5.3: Modelo geral de um processo/sistema
Fonte: Montgomery (2005, p. 2)

Portanto, adotou-se o DOE como arcabouço estruturado para a realização dos experimentos nesta etapa do método. A seguir será fornecida uma descrição sucinta dos principais termos referentes ao DOE, os quais serão usados durante o restante deste trabalho:

- Variável resposta: é uma saída do sistema que tem interesse para o estudo experimental. Uma ou mais variáveis resposta podem ser consideradas no experimento.
- Fatores: são variáveis do sistema, controladas ou não pelo experimentador, que podem afetar a variável resposta. Os fatores podem ser quantitativos ou qualitativos. Estes também podem ser controláveis e não controláveis (em nível experimental às vezes são passíveis de algum tipo de controle).
- Níveis do fator: Os valores que um fator pode assumir durante um experimento.
- Tratamento: Uma particular combinação de níveis dos fatores incluídos no modelo do estudo experimental.
- Ensaio: A execução de um tratamento.
- Experimento: A realização de todos ou partes dos ensaios de um projeto experimental.
- Replicação: A repetição da execução de um experimento.
- Projeto experimental: é o resultado de uma estratégia de ação, também chamada de estratégia de experimentação, que determina o número e a configuração dos tratamentos, a quantidade de replicações e a ordem de execução dos ensaios.

Usualmente, experimentos são conduzidos para determinar a influência que os fatores exercem sobre a variável resposta (MONTGOMERY, 2005). Nesta etapa da proposta, o objetivo é testar um conjunto de fatores candidatos a fim de produzir dados para a avaliação da influência destes fatores sobre os efeitos do envelhecimento de software. Portanto, assume-se que a maioria dos fatores do projeto experimental estará associada aos parâmetros da carga de trabalho do sistema, haja vista que os padrões de ativação das FRE estão, na maioria das vezes, relacionados às requisições submetidas ao sistema.

Com relação aos níveis de cada fator, assume-se que estes serão definidos a partir do conhecimento do experimentador a respeito do sistema investigado. Este conhecimento é o que Montgomery (2005) denomina de conhecimento do processo, sendo um atributo essencial para o experimentador encarregado da execução desta etapa do método proposto. Os resultados obtidos durante a caracterização da carga de trabalho e a identificação e validação do envelhecimento, são subsídios importantes para a definição dos níveis dos fatores.

Sobre a variável resposta (y), a mesma deve ser uma medida do processo de aplicação, ou do sistema operacional, que indique o grau de envelhecimento do SST. Esta definição também tem como base os experimentos prévios, especialmente aqueles usados para confirmar a presença do envelhecimento no SST. Em termos práticos, a maioria dos sistemas operacionais modernos oferece algum tipo de interface para se monitorar os dados de contabilização dos recursos utilizados, tanto por processos, quanto pelo próprio sistema operacional. Como exemplo deste tipo de interface, tem-se o repositório *Registry* nos sistemas operacionais da família Windows e a interface */proc* do sistema operacional Linux.

Com relação à estratégia de experimentação, Freitas Filho (1997) descreve as seguintes abordagens: bom senso (*best-guess*), um fator por vez (*one-factor-at-a-time*), fatorial e suas variações. Nesta etapa, as necessidades de análise são voltadas para a caracterização do envelhecimento de software com respeito aos fatores que mais o influenciam. De acordo com Montgomery (2005), um dos enfoques adequados para estudar os efeitos de vários fatores sobre a variável resposta é conduzir um projeto fatorial, já que neste projeto todos os fatores são modificados juntos ao invés de apenas um no tempo. Esta afirmação corrobora com Neto, Scarminio e Bruns (1995), que também descrevem o uso de projetos fatoriais completos para a avaliação da influência de fatores sobre a variável resposta. Neste caso, considerando a conformidade dos projetos fatoriais com as necessidades desta etapa, estes foram escolhidos para serem aplicados na caracterização do envelhecimento.

Dentre as possíveis configurações de um projeto fatorial, optou-se pelo projeto fatorial 2^k com replicações ($2^k r$), por ser uma configuração particularmente útil em projetos de experimentos voltados para estudos de caracterização (MONTGOMERY, 2005). Nesta abordagem, k representa o número de fatores, τ o número de níveis (neste caso 2) e r o número de replicações. O número total de ensaios (δ) é obtido através da equação 5.1.

$$\delta = \left(\prod_{i=1}^{k} \tau_i \right) \cdot r \tag{5.1}$$

Nos casos onde o resultado seja um número elevado de ensaios, uma alternativa é reduzir o número de fatores adotando projetos fatoriais fracionados. Neste tipo de projeto, a configuração é dada por ($2^{k-p} r$), onde p é o número de fracionamentos escolhido. Este trabalho descreve a utilização dos projetos fatoriais completos do tipo $2^k r$, contudo, uma descrição de como aplicar os procedimentos (ex. tabela de sinais, ANOVA, etc.) que serão descritos a

seguir, voltados para projetos fatoriais fracionados, pode ser obtida no capítulo 8 de Montgomery (2005), capítulo 19 de Jain (1991) e em Barbetta, Reis e Bornia (2004, p. 46).

A fim de ilustrar os conceitos de projeto de experimentos abordados anteriormente, novamente toma-se como exemplo o caso do envelhecimento de software dos *switches Catalyst*. A variável resposta *y* poderia ser a quantidade de memória alocada para o processo `telnetd`, que é acometido pelos efeitos do envelhecimento (aumento gradativo do seu tamanho em memória), causados por uma falta que provoca o vazamento de memória. Com relação aos fatores do projeto experimental, a tabela 5.1 apresenta três exemplos com seus respectivos níveis.

Tabela 5.1: Exemplo de fatores e níveis

Nível	Tipo de requisição	Duração	Resultado
1	TELNET	<= 3 seg	Sucesso
2	Outros	> 3 seg	Falha

O fator tipo de requisição representa o protocolo de conexão com o equipamento, podendo ser tanto TELNET quanto outros protocolos (ex. SSH, FTP) suportados pelo *switch*. O fator duração significa o tempo de sessão de uma conexão. O resultado é um fator que indica o sucesso ou a falha de uma requisição de criação de sessão.

Por se tratar de um projeto fatorial $2^k r$, cada nível de um fator é ensaiado com todos os níveis dos demais fatores (MONTGOMERY, 2005). Portanto, com k=3 fatores ensaiados em dois níveis cada se tem oito tratamentos. A tabela 5.2 lista todos os tratamentos.

Tabela 5.2: Configuração dos tratamentos para o exemplo do *switch*

Tratamento	Tipo de requisição	Duração	Resultado
1	TELNET	<= 3 seg	Sucesso
2	TELNET	> 3 seg	Sucesso
3	TELNET	<= 3 seg	Falha
4	TELNET	> 3 seg	Falha
5	Outros	<= 3 seg	Sucesso
6	Outros	> 3 seg	Sucesso
7	Outros	<= 3 seg	Falha
8	Outros	> 3 seg	Falha

Com relação às replicações destes tratamentos, neste exemplo está sendo considerado que r=1, ou seja, cada tratamento é executado apenas uma única vez. Com base no diagnóstico (CISCO SYSTEMS, 2000) do fornecedor do *switch*, o padrão de ativação da falta

que causa a degradação (envelhecimento) da memória do *switch* é representado por três dos oito tratamentos sugeridos. A tabela 5.3 apresenta estes três tratamentos.

Tabela 5.3: Tratamentos selecionados para os testes de envelhecimento do *switch*

Tratamento	Tipo de requisição	Duração	Resultado
1	TELNET	<= 3 seg	Sucesso
3	TELNET	<= 3 seg	Falha
4	TELNET	> 3 seg	Falha

Para a definição do F_E a partir dos tratamentos da tabela 5.3, seria necessário avaliar qual o grau de influência de cada tratamento, ou da combinação destes, sobre o envelhecimento do processo `telnetd`. Com base nesta avaliação, o F_E seria composto do fator, ou fatores, com maior influência sobre o envelhecimento do `telnetd`. Neste exemplo, o nível de influência de cada tratamento não está disponível nos dados reportados pelo fornecedor, o que exigiria a sua obtenção de forma experimental, etapa que será discutida na seção 5.3.2.

No exemplo descrito anteriormente, a configuração dos tratamentos adotou uma única replicação. De forma geral, no DOE a escolha do número de replicações está associada à necessidade de análise do experimentador. Em muitos casos, tem-se como objetivo realizar um teste de hipótese para detectar diferenças nas médias das estimativas da variável resposta para cada tratamento (MONTEGOMERY, 2005, p. 41). No método proposto, o resultado do DOE será usado para selecionar os fatores com maior influência na variável resposta, ou seja, identificar as combinações de fatores e níveis que melhor representem o F_E. Neste caso, não se pretende testar a diferença entre médias, ou mesmo a diferença destas com algum valor de referência que poderia ser especificado com base em outros trabalhos. Atualmente, não existem estudos que permitam definir qual a magnitude de degradação da variável resposta, que possa ser considerada uma manifestação representativa do envelhecimento de software. Neste sentido, o método baseia-se apenas na estimação dos efeitos causados por cada fator sobre a variável resposta, de forma a selecionar aqueles que possuem a maior influência na degradação do recurso sendo monitorado. Esta seleção tem como base análises quantitativas dos efeitos estimados, a fim de verificar aqueles de maior significância para a composição do ranking que resultará na seleção do F_E. As técnicas para realizar a análise quantitativa dos efeitos serão descritas na próxima seção.

Baseado no exposto anteriormente, o enfoque adotado pelo método para definir o número de replicações foi baseado no intervalo de confiança para as estimativas dos efeitos.

Nesta abordagem, o número de replicações é definido de forma que as estimativas dos efeitos estejam dentro de um intervalo de confiança, cuja margem de erro tolerável não ultrapasse um valor previamente arbitrado. Para tanto, o método adotou a definição de erro-padrão dos efeitos descrita em Montgomery e Runger (2003, p. 317) e representada pela equação 5.2. Vale ressaltar que em (5.2) a estimativa do erro-padrão do efeito foi dividida por dois, haja vista que o método proposto trabalha com a metade das estimativas dos efeitos, como será descrito em detalhes na próxima seção.

$$ep(efeito) = \frac{1}{2}\sqrt{\frac{1}{n2^{k-2}}\sigma^2} \tag{5.2}$$

O intervalo de confiança para o efeito é definido em (5.3).

$$\frac{\bar{y}_+ - \bar{y}_-}{2} \pm t_{\alpha/2,glerro} \cdot ep(efeito)\,, \tag{5.3}$$

onde $(\bar{y}_+ - \bar{y}_-)$ é o efeito de um dado fator/interação.

A partir de (5.3) e da especificação do erro amostral tolerado (E_0), o número de replicações para o DOE é obtido resolvendo a equação 5.4:

$$t_{\alpha/2,glerro} \cdot \frac{1}{2}\sqrt{\frac{1}{n2^{k-2}}\sigma^2} = E_0\,, \tag{5.4}$$

onde:

$t_{\alpha/2,glerro}$ é o valor tabelado da distribuição *t* de *Student* para um dado nível de significância (α) e 2^k(n-1) graus de liberdade do erro;

E_0 é um valor arbitrado na mesma unidade da variável resposta do DOE;

σ^2 assume o valor da variância da amostra piloto (s_0^2).

Assumindo-se $t_{\alpha/2,glerro} \approx 2$ e isolando *n*, tem-se:

$$n = \frac{\sigma^2}{E_0^2 \cdot 2^{k-2}} \tag{5.5}$$

Para o cálculo de *n* o método necessita de uma amostra piloto a fim de obter uma estimativa para σ^2. Esta amostra é obtida executando-se um tratamento com todos os fatores definidos com o valor médio dos seus níveis. Como o método considera apenas dois níveis, o

valor de cada fator será o valor médio entre os seus níveis alto e baixo. Neste caso, a amostra piloto é obtida a partir de vinte replicações deste tratamento. Como resultado de (5.5), o valor de *n* representa o número de replicações (*r*) da equação 5.1.

Outro aspecto importante no projeto de experimentos é a aleatorização dos tratamentos. Na maioria dos experimentos em envelhecimento de software, é possível se ter para cada ensaio o mesmo ambiente de teste, já que os elementos envolvidos tratam-se de componentes de software. Como exemplo, no caso do *switch*, antes de cada ensaio a reinicialização (*reboot*) do equipamento garante o mesmo ambiente de execução (ex. mesma quantidade de memória disponível, mesmo código carregado em memória, etc.) para todos os ensaios. Para os casos onde o padrão de ativação das FRE, ou o efeito do envelhecimento, sejam dependentes de algum recurso que não possa ser reinicializado para o mesmo estado antes da execução de cada ensaio, nestes casos se faz necessário adotar a aleatorização da alocação dos tratamentos em cada unidade experimental (MONTGOMERY, 2005).

Todos os aspectos do projeto experimental, citados anteriormente, devem ser conduzidos de forma disciplinada a fim de garantir a confiabilidade dos seus resultados. Um trabalho que aborda esta importância, bem como sugere vários formulários para o registro das etapas, é descrito em Coleman e Montgomery (1993). O quadro 5.1 lista os passos sugeridos neste trabalho.

Passo	Descrição
1	Reconhecimento e declaração do problema
2	Escolha dos fatores e níveis*
3	Seleção da variável resposta*
4	Escolha da estratégia de experimentação
5	Execução do experimento
6	Análise dos dados
7	Conclusões e recomendações
* Em algumas situações práticas os passos 2 e 3 podem ser invertidos.	

Quadro 5.1: Procedimentos para experimentação
Fonte: Coleman e Montgomery (1993)

De forma geral, esta etapa de caracterização do envelhecimento de software baseia-se nestes passos. O passo 1 é representado pelo estudo de identificação do envelhecimento de software. O passo 2, em grande parte, refere-se à caracterização da carga de trabalho do sistema analisado. Como já discutido no início da seção 5.3, os passos 1 e 2 são resultados de estudos prévios à esta etapa. O passo 3 é apoiado nos resultados dos passos 1 e 2, levando-se em conta a compatibilidade desta escolha com os procedimentos que serão adotados para a medição da degradação (envelhecimento) do sistema durante a execução do ADT (ver seção

5.4). O passo 4 foi definido nesta seção, sendo adotado o projeto fatorial $2^k r$. A execução do experimento (passo 5) é específica ao tipo de sistema analisado, ambiente de teste, instrumentação de medição, etc. Finalmente, os passos 6 e 7 serão descritos na próxima seção.

5.3.2. Análise dos efeitos dos fatores sobre o envelhecimento de software

A partir dos valores amostrais de *y*, obtidos na etapa anterior, serão selecionados como F_E os fatores que exerçam maior influência sobre o envelhecimento do sistema analisado. Esta seleção é baseada na computação dos efeitos de cada fator, assim como de suas interações, sobre a variável resposta. Como resultado, um ranking dos fatores é construído, tendo como regra de classificação os seus efeitos sobre *y*. O restante desta seção apresenta uma descrição detalhada de cada um destes procedimentos.

Primeiramente, o método prevê a realização de uma análise de variância (ANOVA), acompanhada de um teste *F*, com o objetivo de verificar quais são os fatores cujos efeitos, no envelhecimento dos processos, são significativos para serem considerados no ranking para a seleção do F_E. Para tanto, recomenda-se a utilização de um teste *F* com nível de significância $\alpha = 0{,}05$. Os procedimentos para a realização da ANOVA, para projetos de experimentos, são amplamente descritos pela literatura e, portanto, não serão apresentados neste trabalho. Uma descrição detalhada de sua utilização em projetos fatoriais do tipo 2^k pode ser encontrada em Barbetta, Reis e Bornia (2004, p.263) e Montgomery (2005, p.167). Também, no estudo experimental apresentado no capítulo 6 (ver seção 6.4.1.2) tem-se um exemplo da aplicação da ANOVA durante a utilização do método proposto.

A partir dos resultados da ANOVA, tem-se a identificação dos fatores que serão avaliados quantitativamente para a seleção do F_E. A próxima etapa é quantificar os efeitos de cada fator sobre a variável resposta (*y*), com o objetivo de determinar a fração da variação de *y* explicada por cada fator considerado nesta etapa. Para tanto, adota-se o método da tabela de sinais (FREITAS FILHO, 2001, p. 266; JAIN, 1991, p. 286), também chamada tabela de sinais + e – (MONTGOMERY, 2005, p. 208). Segundo Neto, Scarbini e Bruns (1995) este é um "[...] procedimento que permite calcular qualquer efeito sem dificuldade, não importa o tamanho do planejamento". Tal afirmação corrobora com Jain (1991, p. 294), que descreve este método como uma forma simples de analisar um projeto $2^k r$. A seguir uma síntese do algoritmo usado para o cálculo dos efeitos dos fatores a partir da tabela de sinais, de acordo com a descrição em Jain (1991, p. 286).

5.3.2.1. Método da tabela de sinais

O primeiro passo é a construção da matriz de sinais que será usada para a computação dos efeitos. Os valores correspondentes aos níveis dos fatores são codificados em -1 ou +1. Um exemplo desta matriz é a tabela 5.4, a qual foi construída para um projeto fatorial 2^2 com 3 replicações. Vale ressaltar que o algoritmo descrito nesta seção vale para qualquer projeto fatorial $2^k r$.

Tabela 5.4: Matriz de sinais para a computação dos efeitos de um projeto $2^2 \cdot 3$

I	*A*	*B*	*AB*	R1	R2	R3	$\bar{y}_{ij}$
1	-1	-1	1	y_{11}	y_{12}	y_{13}	$\bar{y}_{1.}$
1	1	-1	-1	y_{21}	y_{22}	y_{23}	$\bar{y}_{2.}$
1	-1	1	-1	y_{31}	y_{32}	y_{33}	$\bar{y}_{3.}$
1	1	1	1	y_{41}	y_{42}	y_{43}	$\bar{y}_{4.}$
$\bar{y}_{..}$	*ef*(*A*)	*ef*(*B*)	*ef*(*AB*)				

A primeira coluna (I) contém o valor +1 em todas as linhas. As próximas colunas são referentes aos efeitos principais (*A* e *B*) e suas interações. Neste projeto houve apenas uma interação (*AB*), diferente de um projeto 2^3, por exemplo, que possuiria 4 interações. Posterior às colunas de interações, se tem as colunas referentes ao valor de *y* para cada replicação. A última coluna contém a média aritmética dos valores da variável resposta (y_{ij}), observados em cada replicação, a qual será usada como referência para a computação dos efeitos. A notação de ".", usada nos subscritos da variável da última coluna, implica no somatório do subscrito ao qual o "." se aplica. Por exemplo, $\bar{y}_{1.}$ representa a média das replicações de *y* para o primeiro tratamento, ou seja:

$$\bar{y}_{1.} = \frac{y_{1.}}{n},$$

onde *n* representa o número de replicações do tratamento 1.

A configuração apresentada na tabela 5.4 lista a combinação de tratamentos na chamada ordem padrão ou ordem de Yates (MONTGOMERY, 2005, p. 207). Todas as colunas dos fatores individuais começam com o nível (-) e depois os sinais vão se alternando, um a um na primeira coluna (- + - +), dois a dois na segunda coluna (- - + +) e assim por diante. Com um

total de k fatores, se têm na última coluna de fatores individuais 2^{k-1} sinais negativos e 2^{k-1} sinais positivos. Os sinais das colunas de interações são obtidos multiplicando os sinais das colunas que correspondem ao nome da interação. Por exemplo, no caso da interação *AB*, multiplica-se os sinais da coluna *A* com os sinais da coluna *B*. A última linha é dedicada aos resultados do cálculo dos efeitos.

O próximo passo é computar os efeitos dos fatores e interações. Para isso, assumindo cada coluna da matriz como um vetor, o efeito de cada fator é o resultado do produto escalar do vetor de cada fator pelo vetor de respostas (última coluna), dividido pelo número de tratamentos[26]. Como exemplo, para calcular o efeito do fator *A* (ver tabela 5.4) tem-se:

$$ef(A) = \left(\frac{1}{4}\right) \cdot \left\{ \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \bar{y}_{1.} \\ \bar{y}_{2.} \\ \bar{y}_{3.} \\ \bar{y}_{4.} \end{bmatrix} \right\} \tag{5.6}$$

Depois de computados todos os efeitos, estes devem ser transportados para a última linha das respectivas colunas de cada fator, juntamente com o valor médio do experimento que é colocado na primeira coluna desta linha (JAIN, 1991, p. 286). Uma representação genérica deste cálculo segue. Os vetores estão representados tipograficamente em negrito:

$$ef(F_x) = \left(\frac{1}{n}\right) \cdot \mathbf{F}_x^t \cdot \mathbf{Y} \,, \tag{5.7}$$

onde:

x representa cada fator e suas interações, incluindo a média global (coluna I);

$\mathbf{F}_x$ é o vetor coluna de x;

$\mathbf{F}_x^t$ é o vetor linha obtido pela transposição do vetor coluna $\mathbf{F}_x$;

$\mathbf{Y}$ é o vetor de respostas, o qual possui as médias das replicações de cada tratamento;

n é o número de tratamentos (2^k).

A fim de ilustrar a interpretação destes procedimentos, a tabela 5.5 apresenta a matriz de sinais do exemplo 17.1 descrito em Jain (1991), para estudar o impacto do tamanho da memória RAM (*megabytes*) e *cache* (*kilobytes*) sobre a performance de uma estação de trabalho sendo projetada. Os fatores memórias RAM e *cache* serão representados

[26] Neste caso quatro.

respectivamente pelas letras *A* e *B*. A variável resposta (*y*) representa a performance da estação de trabalho em MIPS. Neste exemplo cada tratamento foi executado uma única vez.

Tabela 5.5: Matriz de sinais do projeto da estação de trabalho
Fonte: Jain (1991, p. 284)

I	*A*	*B*	*AB*	*y*
1	-1	-1	1	15
1	1	-1	-1	45
1	-1	1	-1	25
1	1	1	1	75
40	20	10	5	

Jain (1991, p. 285) demonstra que o algoritmo para o cálculo dos efeitos baseado na matriz de sinais, apresentado anteriormente, computa os efeitos considerando-os como coeficientes de uma equação de regressão. Um exemplo desta representação é a equação 5.8 definida com base nos dados da tabela 5.5.

$$y = 40 + 20x_a + 10x_b + 5x_a x_b \tag{5.8}$$

Neto, Scarminio e Bruns (1995, p. 75) explicam que neste algoritmo, devido à codificação dos verdadeiros valores dos níveis de cada fator, cada efeito passa a corresponder sempre à variação de *duas* unidades do fator correspondente, já que o nível varia de –1 para +1. Isso quer dizer que os efeitos *por unidade* de x_a e x_b são a metade dos efeitos calculados sem codificação, explicando porque a variável *n* em (5.7) é 2^k e não 2^{k-1}.

Desta forma, a interpretação dos efeitos (coeficientes de 5.8) calculados para o exemplo da estação de trabalho, segue:

- o valor médio da performance da estação de trabalho é de 40 MIPS;
- o efeito da memória RAM (x_a) sobre a performance média da estação é de 20 MIPS;
- o efeito da memória *cache* (x_b) sobre a performance média da estação é de 10 MIPS;
- a interação entre as memórias *cache* e RAM causam um efeito de 5 MIPS na performance média do sistema.

O próximo passo do método proposto é construir um ranking dos fatores, onde a classificação é baseada na contribuição estimada, de cada fator, sobre as variações do envelhecimento de software. A importância de cada fator é medida pela proporção da variação estimada, da variável resposta, que é explicada pelo fator. A partir destes valores, um

ranking é construído e a seleção do F_E pode ser realizada. A seguir os passos para a construção do ranking.

Primeiramente, calcula-se a variação total da variável resposta. Este valor é conhecido como soma dos quadrados totais (*SQT*) (FREITAS FILHO, 2001, p. 269; JAIN, 1991, p. 286), sendo obtido através da equação 5.9.

$$SQT = \Sigma_{i=1}^{2^k} \left(\bar{y}_{i.} - \bar{y}_{..} \right)^2 \tag{5.9}$$

No exemplo da estação de trabalho, o resultado seria:

$$SQT = \Sigma_{i=1}^{2^k} \left(\bar{y}_{i.} - \bar{y}_{..} \right)^2 = (15-40)^2 + (45-40)^2 + (25-40)^2 + (75-40)^2$$

$$= \left(625 + 25 + 225 + 1225\right) = 2100$$

A *SQT* pode ser expressa como a soma das porções da variação total explicada pelos efeitos de cada fator e interações. No exemplo da estação de trabalho, a *SQT* pode ser representada como:

$$SQT = SQA + SQB + SQAB \tag{5.10}$$

A fim de facilitar a análise e seleção do F_E, a construção do ranking considera os valores das contribuições expressos em termos percentuais. De acordo com Jain (1991, p. 287), esta representação facilita a avaliação da importância de cada fator. Portanto, considerando o exemplo da estação de trabalho, a fração da variação explicada pelo fator *A* é dada por:

$$\textit{Fração da variação explicada pelo fator } A = \frac{SQA}{SQT} \ , \tag{5.11}$$

onde *SQA* é calculado como segue:

$$SQA = ef(A)^2 \cdot 2^k \tag{5.12}$$

Resumindo os passos descritos anteriormente, a tabela 5.6 e o quadro 5.2 apresentam um exemplo numérico dos procedimentos para a construção do ranking. Os dados usados foram do exemplo da estação de trabalho.

Tabela 5.6: Ranking dos fatores ordenado pela contribuição na variação de *y*

Fator/Interações	Contribuição na variação total de *y*	Ranking
A	76%	1º
B	19%	2º
AB	4%	3º

$SQA = ef(A)^2 \cdot 2^k = (20)^2 \cdot 4 = 1600$	*Fração da variação explicada* $A = \frac{SQA}{SQT} = \frac{1600}{2100} = 0{,}76$
$SQB = ef(B)^2 \cdot 2^k = (10)^2 \cdot 4 = 400$	*Fração da variação explicada* $B = \frac{SQB}{SQT} = \frac{400}{2100} = 0{,}19$
$SQAB = ef(AB)^2 \cdot 2^k = (5)^2 \cdot 4 = 100$	*Fração da variação explicada* $AB = \frac{SQAB}{SQT} = \frac{100}{2100} = 0{,}04$

Quadro 5.2: Cálculo das contribuições de cada fator sobre a variação da variável resposta

Este exemplo não tem relação com o problema do envelhecimento de software, contudo, utilizando-o para ilustrar a utilização do ranking, o fator *A* poderia ser escolhido para ser usado como F_E, haja vista sua maior contribuição para a variação total da variável resposta. Vale destacar que poderiam ter dois ou mais fatores com contribuições relevantes, como, por exemplo, A (50%) e B (45%). Nestes casos, o F_E poderia ser representado por ambos fatores na carga de trabalho.

A figura 5.4 apresenta os principais elementos discutidos nesta seção.

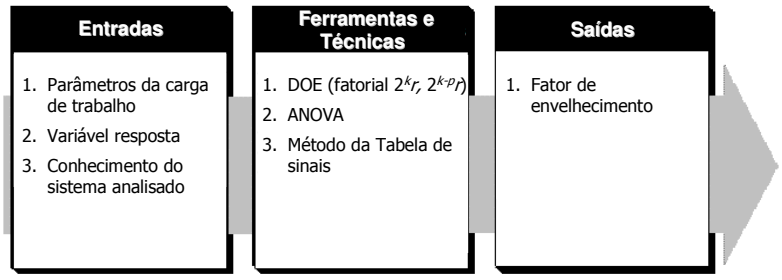

Figura 5.4: Principais elementos do 1º processo

5.4. TESTE DE ENVELHECIMENTO ACELERADO

Após a seleção do fator de envelhecimento, o próximo processo a ser executado é o teste de envelhecimento acelerado (TEA). Como definido na seção 5.1, o método proposto

adota uma abordagem baseada em ADT. Neste caso, o termo degradação foi substituído por envelhecimento, sendo mais adequado ao contexto deste trabalho, o que deu origem ao termo TEA. Assim como para a seleção do F_E, a execução do TEA exige uma etapa de planejamento, a qual foi detalhada na seção 4.3. A seguir, cada elemento que compõe este planejamento será apresentado para o método proposto.

5.4.1. Forma do teste de envelhecimento acelerado

A seguir, os elementos de definem a forma do TEA são apresentados.

5.4.1.1. Medida de performance

Durante o TEA, a degradação do sistema é medida periodicamente a fim de se construir os caminhos de degradação a serem utilizados na obtenção dos tempos de *pseudofalha*. Normalmente, esta medida será a mesma que foi utilizada como variável resposta (y) durante a seleção do F_E.

A construção dos caminhos de degradação (ver seção 4.2.6.1) ocorre tomando-se periodicamente os valores da variável de performance. Quanto ao número de medidas que devem ser realizadas, os trabalhos na área não apresentam um padrão para este valor. Por exemplo, em Meeker e Escobar (1998, p. 338) foi utilizado um ADT com 17 medições, mais do que o dobro em relação ao valor utilizado em Nelson (1981), que foi de 8 medições. Diferente dos exemplos anteriores, Carey e Koeng (1991) realizaram um número específico de medições para cada nível de estresse (9 - baixo, 7 - intermediário e 13 - alto).

Ressalta-se que os caminhos de degradação são séries de dados temporais que, na maioria dos casos, serão usadas para a predição dos tempos de *pseudofalha*. Portanto, técnicas apropriadas para a previsão de dados em séries temporais devem ser empregadas para a obtenção destes tempos. Neste contexto, aspectos como o tipo de curva (ex. linear, curvilínea, etc.), a técnica de estimação dos parâmetros do modelo, o intervalo de confiança das estimativas, restrições práticas (ex. mecanismo de medição), dentre outros aspectos, estão diretamente relacionados com as decisões que determinarão a quantidade de medições usadas na construção dos caminhos de degradação.

A fim de se manter genérico quanto à forma do caminho de degradação, o método deixa a critério do experimentador a escolha da abordagem a ser empregada para a obtenção dos tempos de *pseudofalha*. Conseqüentemente, o número de medições que serão realizadas irá

depender da abordagem adotada. Em Hanke, Reitsch e Wichern (2001) e Neter *et al.* (1996) são apresentadas várias técnicas usadas para este propósito.

5.4.1.2. Determinação das condições de teste

A carga de trabalho usada durante o TEA é a mesma caracterizada nas condições normais de operação do SST, com exceção dos valores atribuídos ao(s) fator(es) selecionado(s) para F_E. Um aspecto importante que determina a confiabilidade dos resultados do TEA é a capacidade de reprodução, em nível experimental, das mesmas condições de operação existentes no ambiente real. Em muitos casos, o mesmo ambiente utilizado para executar os experimentos da etapa de seleção do F_E servirá para a implementação do TEA.

5.4.1.3. Definição das variáveis de estresse

O conceito de variável de estresse ou estresse de aceleração foi definido no início do capítulo, como os padrões da carga de trabalho que ativam as faltas relacionadas ao envelhecimento. As variáveis de estresse que serão usadas no TEA são representadas pelo fator de envelhecimento (F_E), que foi obtido com a execução do primeiro processo do método. O F_E é considerado como uma das entradas do atual processo.

5.4.1.4. Definição da forma de aplicação da carga de estresse

Dentre as formas de aplicação estudadas no capítulo 4, o método proposto é baseado na utilização de estresse constante. Esta escolha está fundamenta nos prós e contras de cada forma de aplicação discutidos na seção 4.2.4. A adoção desta forma de estresse é predominante na literatura, corroborando com os argumentos de Nelson (2004, p. 19) sobre a maior maturidade desta abordagem.

Nesta configuração, para cada nível de estresse o TEA adota a mesma composição da carga de trabalho caracterizada para o nível de uso, com exceção dos valores das requisições do tipo F_E, que são diferentes para cada nível de estresse. Neste caso, a taxa de chegada do F_E é a mesma para todos os níveis. Contudo, pode ser necessário adotar uma taxa de chegada do F_E, diferente daquela caracterizada para o nível de uso, principalmente quando a ocorrência do F_E for considerada um evento raro. Para estas situações, ressalta-se a importância de verificar a suposição de que, o aumento na taxa de uso, não exerça influência sobre os efeitos

do envelhecimento (ver seção 4.2.3.1), já que se está trabalhando com estresse constante. Se comprovado este pressuposto, a taxa de chegada do F_E pode ser controlada de forma a se reduzir o tempo de execução do TEA.

Ao se usar uma taxa de chegada do F_E, diferente daquela caracterizada para o nível de uso, exige-se ao final do TEA que as estimativas (ex. MTTF) sejam convertidas. Por exemplo, considere um cenário onde a taxa de chegada do F_E, caracterizada para o nível de uso, tenha sido de 1 ocorrência a cada 10 horas. Durante o TEA, foi utilizada uma taxa de 1 ocorrência a cada hora para todos os níveis de estresse, respeitando o pressuposto descrito anteriormente. Neste cenário, a partir dos resultados do TEA, obteve-se uma estimativa de MTTF de 20 horas para o nível de uso. Devido à utilização de uma taxa de chegada diferente daquela caracterizada para o nível de uso, necessita-se compatibilizar o valor do MTTF com a taxa de chegada caracterizada para o nível de uso. Para este exemplo, a transformação aplicável deve ser MTTF x 10 = 200 horas.

5.4.1.5. Mecanismo de censura

Como o TEA é um teste do tipo ADT, se faz necessário definir a duração do teste (D_t) e o limiar de envelhecimento (D_f). Um valor de D_t ideal seria aquele em que todos os caminhos de degradação alcançassem o limiar D_f durante a execução do TEA. Contudo, na prática a obtenção de falhas de envelhecimento nos níveis mais baixos de estresse (próximos ao nível de uso), normalmente exige ensaios por longos períodos ininterruptos, o que muitas vezes não é possível devido a restrições de tempo e recursos. A definição do valor D_t deve ter como base estas restrições, bem como o conhecimento do experimentador sobre o comportamento dos efeitos do envelhecimento sobre o SST. A etapa prévia de validação da existência do envelhecimento pode ser usada como referência.

Com relação ao valor de D_f, este deve ser um limiar que, ao ser atingido ou ultrapassado pela medida de performance, se declara que ocorreu uma falha do sistema. A condição de falha ($y \geq D_f$ ou $y \leq D_f$) depende do tipo de medida de performance adotada. Por exemplo, se a medida de performance é o tamanho de um processo e a falha ocorre quando este processo atingir um determinado tamanho (ex. D_f = 1 megabytes), então a condição de falha será $y \geq D_f$. De outra forma, y poderia ser a taxa de resposta de um sistema que falha quando sua performance for abaixo de D_f, neste caso a condição de falha é dada por $y < D_f$.

A fim de melhor compreender o comportamento do envelhecimento, com o objetivo de definir os valores mais adequados para D_t e D_f, as etapas de identificação do envelhecimento e seleção do F_E são importantes fontes de informação para o experimentador.

5.4.2. Plano experimental

Dentre as três propostas de plano experimental (ver seção 4.3), sugere-se inicialmente adotar planos tradicionais ou de compromisso. Os planos tradicionais geralmente apresentam desempenho inferior aos demais em termos de acuracidade das estimativas. Contudo, Nelson (2004, p. 349) explica que em casos onde são necessárias extrapolações maiores, os planos tradicionais com três níveis e igual alocação podem oferecer maior acuracidade e tolerância a desvios do modelo de relacionamento.

Os planos de compromisso, assim como os planos tradicionais, por possuírem mais de dois níveis, são mais tolerantes aos possíveis desvios do modelo estresse-envelhecimento. Contudo, a política de alocação também é um aspecto importante a se considerar. Nelson (2004, p. 349) avalia a alocação 4:2:1, dos planos de Meeker-Hahn, como sendo muito desigual, sendo mais apropriada para os casos onde a exigência de extrapolação é pequena. O mesmo autor sugere, como melhor alocação de compromisso, um maior número de testes nas extremidades (sendo maior no nível inferior), com o nível intermediário recebendo a menor parcela. Desta forma estes planos se aproximam dos planos tradicionais em condições que envolvam um maior grau de extrapolação.

Os planos ótimos, apesar de proporcionarem maior acuracidade com menos testes, possuem a limitação de terem apenas dois níveis. Segundo Meeker e Escobar (1998, p. 544), por causa desta característica os planos ótimos tendem a possuir pouca robustez frente a desvios do modelo de relacionamento estresse-envelhecimento.

A sugestão inicial de se usar planos tradicionais ou de compromisso, considera basicamente os aspectos discutidos na revisão da literatura, principalmente o melhor desempenho destes frente a cenários envolvendo extrapolações mais longas e eventuais desvios do modelo de relacionamento. A ausência de trabalhos com resultados sobre a aplicação destes planos em ensaios acelerados para produtos de software, motivou a proposta de se usar planos mais conservadores neste estágio da pesquisa.

Para concluir a etapa de planejamento do TEA, é necessário definir o número de testes (tamanho da amostra) que serão executados de acordo com os planos sugeridos.

5.4.2.1. Cálculo do tamanho da amostra do TEA

Considerando que além das duas sugestões anteriores o experimentador tenha a necessidade de implementar novos tipos de planos, optou-se por um procedimento que seja aplicável aos três planos experimentais e para qualquer política de alocação. Este procedimento está descrito em Nelson (2004, p. 327) e considera os seguintes pressupostos:

a) Todos os testes falham durante o TEA;
b) O relacionamento vida-estresse seja do tipo linear;
c) A distribuição dos tempos de falha segue um dos seguintes modelos: Lognormal, Weibull ou Exponencial;
d) As estimativas de interesse são a média ou mediana dos tempos de falha;
e) A estimação dos parâmetros é realizada pelo método dos mínimos quadrados.

A suposição (a) é satisfeita considerando que no TEA, para cada caminho de degradação, se tem um tempo de falha ou *pseudofalha*. Com relação ao modelo de relacionamento (b), será adotado o IPL, o qual é *linearizável* (ver seção 4.2.5.1.1). Com respeito ao pressuposto (c), a distribuição de Weibull foi usada por Garg *et al.* (1996a) para modelar os tempos de falha por envelhecimento de software. Ainda, Nelson (2004, p. 170) destaca que uma das aplicações da distribuição Lognormal é para modelar dados de degradação, como é o caso dos testes de envelhecimento acelerados (TEA).

O procedimento adotado para o cálculo da amostra está exposto no apêndice A. Relacionado ao item (e), Nelson (2004, p. 190) explica que quando a distribuição de vida for Weibull ou Exponencial, o método da máxima verossimilhança pode ser usado, pois oferece estimativas mais acuradas do que o método dos mínimos quadrados. Esta abordagem está descrita em detalhes em Nelson (2004, p. 330) e Freitas e Colosimo (1997, p. 240). Para os casos onde o MLE for adotado, uma sugestão prática é que no mínimo existam 20 tempos de falhas (NELSON, 2004, p. 236). Baseando-se neste valor e considerando a adoção dos planos tradicionais ou de compromisso, com três níveis, o método considera um número mínimo de 10 testes para cada nível, totalizando 30 falhas/*pseudofalhas* ao final do TEA.

5.4.3. Obtenção dos tempos de falha/pseudofalha

Nesta etapa, são executados os testes de envelhecimento acelerados, de acordo com o planejamento da etapa anterior. A duração de cada teste segue a política anteriormente descrita, baseada em D_t e D_f.

O resultado de cada execução é um caminho de degradação com os dados do envelhecimento, obtidos por meio da mensuração da medida de performance que indica o grau de envelhecimento do sistema durante o TEA.

Para os testes em que o caminho de degradação não atinja o limiar D_f até o tempo D_t, adota-se os quatro primeiros procedimentos da técnica de análise de degradação acelerada aproximada (ver seção 4.2.6.2), com o objetivo de realizar a predição dos tempos de *pseudofalha*.

Como resultado desta etapa, tem-se os tempos de falha/*pseudofalha* para todos os níveis de estresse, constituindo a saída do processo TEA. A figura 5.5 apresenta os principais elementos discutidos nesta seção.

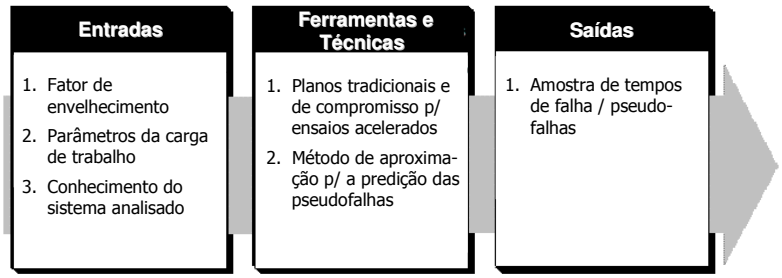

Figura 5.5: Principais elementos do 2º processo

5.5. MODELAGEM DO RELACIONAMENTO ESTRESSE-ENVELHECIMENTO ACELERADO

Esta etapa é basicamente a implementação do último procedimento da técnica de análise de degradação acelerada aproximada, considerada no processo anterior para os casos onde se faz necessário realizar a predição dos tempos de *pseudofalhas*.

5.5.1. Seleção da distribuição dos tempos de falha/*pseudofalha*

Como já abordado no capítulo 4, um pressuposto para a aplicação do método é de que os tempos de falha/*pseudofalha*, obtidos em cada nível de estresse, sigam a mesma distribuição de probabilidades. Outra suposição é que o parâmetro de escala σ, da CDF Φ (ver seção 4.2.5.2), se mantenha constante entre os níveis de estresse.

Primeiramente, deve-se verificar qual distribuição de probabilidades melhor se ajusta aos dados referentes a cada nível de estresse. Diversas técnicas estão disponíveis para este

propósito e basicamente são divididas em duas abordagens: técnicas gráficas e analíticas ou também chamadas testes de aderência/adequação (*Goodness of fit tests*). Os livros Meeker e Escobar (1998), Freitas e Colosimo (1997), Nelson (2004) e Freitas Filho (2001) abrangem ambos enfoques de forma detalhada. Além disso, vale dizer que atualmente estes testes são implementados por diversos pacotes de software, como por exemplo: Statistica[27], MINITAB[28], @Risk Bestfit[29], Weibull++[30], SYSTAT[31], R Project[32], dentre outros.

O método proposto não exige a aplicação de uma técnica específica, ficando a cargo do experimentador esta escolha. Como sugestão, recomenda-se a utilização de ambos enfoques de forma complementar. Neste caso, inicialmente são utilizados, por exemplo, gráficos de probabilidade para avaliar cada distribuição e, posteriormente, se realiza uma avaliação analítica usando testes de aderência, como por exemplo os testes Qui-quadrado (χ^2) ou Kolmogorov-Smirnov (K-S) (FREITAS FILHO, 2001, p. 172; TRIVEDI, 2001, p. 714-724). A seguir um exemplo destes procedimentos. A tabela 5.7 apresenta dados simulados para dois níveis de estresse (S1 e S2) em um sistema hipotético.

Tabela 5.7: Tempos de falha simulados

Tempos de Falha (hr)	
S1=50ºC	S2=100ºC
37,53	16,53
32,89	16,26
33,54	17,50
35,41	16,26
34,67	16,76
35,35	15,98
34,30	16,28
35,29	17,51

A partir destes dados, realizou-se um teste de aderência K-S para avaliar o ajuste de três distribuições (Lognormal, Weibull e Exponencial). O Qui-quadrado não foi usado devido ao tamanho reduzido da amostra (FREITAS FILHO, 2001, p. 174). Os valores da tabela 5.8 foram obtidos com o apoio do software @Risk Bestfit. Para a distribuição Weibull, no nível S2, o software não realizou o teste acusando invalidade do modelo. Neste caso, uma opção natural seria adotar o segundo melhor ajuste, que foi da Lognormal. De acordo com a tabela

[27] www.statsoft.com
[28] www.minitab.com
[29] www.palisade.com
[30] www.reliasoft.com
[31] www.systat.com
[32] www.r-project.org

de valores críticos para a estatística K-S (TRIVEDI, 2001, p. 797), com um nível de significância α=0,05, obteve-se d_8 = 0,45 e, deste modo, não existem evidências suficientes para rejeitar a hipótese nula, portanto aceita-se a Lognormal.

Tabela 5.8: Resultado do teste K-S

Modelo	Nível	Estatística K-S[33]
Lognormal	S1	0,2163
Weibull	S1	0,1820
Exponencial	S1	0,3167
Lognormal	S2	0,2426
Weibull	S2	N/A
Exponencial	S2	0,2995

A figura 5.6 apresenta um gráfico múltiplo de probabilidades (MEEKER; ESCOBAR, 1998, p. 496), onde se tem a distribuição Lognormal ajustada para ambos os níveis de estresse. Neste gráfico foram traçados os intervalos de confiança, neste caso de 90%, a fim de apoiar na verificação do ajuste do modelo. Uma ampla descrição sobre técnicas gráficas para avaliar o ajuste de distribuições, juntamente com os detalhes para a construção destes gráficos, pode ser obtida no capítulo 4 de Nelson (2004) e capítulos 6 e 11 de Meeker e Escobar (1998).

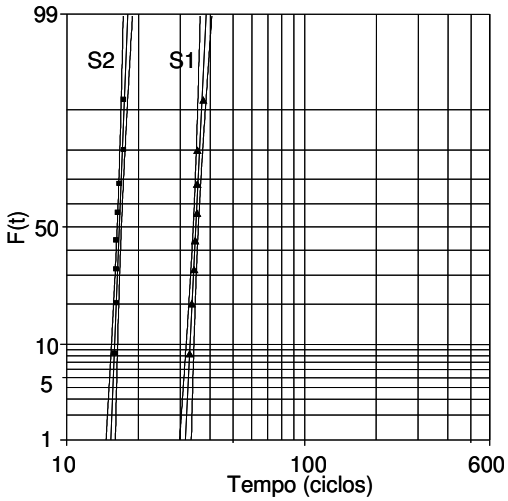

Figura 5.6: Gráfico de probabilidade Lognormal

Ainda relacionado à seleção da distribuição de vida, um aspecto importante é validar a suposição de que o parâmetro σ, de cada CDF ajustada, seja o mesmo para todos os níveis de estresse. Como já discutido anteriormente, não é incomum que estes valores sejam diferentes,

[33] Nesta estatística (distância K-S) quanto menor melhor.

mesmo quando o modelo sendo testado seja adequado e todos os pressupostos necessários estejam sendo atendidos. Neste caso, é fundamental calcular o intervalo de confiança deste parâmetro em cada nível, a fim de se poder avaliar se os valores estão dentro de uma mesma faixa.

O método proposto sugere a utilização de intervalos com nível de confiança de 90% a 95%. Com isso, nos casos onde o σ estimado não for o mesmo em todos os níveis, mas esteja em uma faixa de valores comum a estes, o método recomenda adotar o valor médio desta faixa como referência para a distribuição de vida no nível de uso. Outra abordagem que o método sugere é adotar valores dentro do mesmo IC, os quais estejam próximos do valor de σ estimado para o nível de estresse mais próximo do nível de uso. Como exemplo, a tabela 5.9 apresenta os valores dos parâmetros da Lognormal selecionada anteriormente.

Tabela 5.9: IC dos parâmetros da Lognormal estimado para S1 e S2

Estresse	Parâmetros	Estimativa (MLE)	IC de 90%	
			LI	LS
S1	μ_1	3,55	3,52	3,57
	σ_1	0,03	0,02	0,06
S2	μ_2	2,81	2,79	2,83
	σ_2	0,03	0,02	0,05

Neste exemplo, os parâmetros foram estimados pelo método da máxima verossimilhança. Coincidentemente, os valores de σ foram os mesmos nos dois níveis. Assumindo que os valores fossem $\hat{\sigma}_1$=0,04, LI_1=0,03 e LS_1=0,08, o método proposto sugere adotar um valor de $\hat{\sigma}_1$ que estivesse entre 0,03 e 0,05, sendo este o intervalo de valores de σ comum aos dois níveis.

5.5.2. Relacionamento estresse-envelhecimento

O próximo passo é a definição de um modelo de relacionamento vida-estresse para a projeção da distribuição de vida, selecionada no passo anterior, para o nível de estresse nas condições de uso. Como abordado na seção 4.2.5, os atuais modelos de relacionamento têm como base fenômenos físicos que são comuns nas demais áreas de engenharia e onde ALT/ADT são mais utilizados. A partir da análise dos modelos descritos em Nelson (2004) e Meeker e Escobar (1998), verificou-se que o modelo IPL, apesar de atualmente ser mais usado em ensaios onde a variável de estresse representa voltagem, também se aplica a outras

variáveis de estresse desde que assumam valores positivos e não sejam de natureza térmica. Além disso, o IPL por ser linearizável se enquadra no modelo de regressão linear simples descrito na seção 4.2.5.2, o qual é um dos requisitos para a aplicação do método proposto.

Estas características tornam o IPL, dentre todos os modelos revisados, o mais aderente ao uso em ensaios acelerados envolvendo o envelhecimento de software. Haja vista a ausência de literatura com resultados da aplicação do IPL para a aceleração de falhas de software, uma hipótese desta pesquisa é que o IPL pode ser usado para modelar a relação entre o tempo de falha por envelhecimento de software e os níveis de aceleração atribuídos ao fator de envelhecimento (F_E). Com base nesta suposição, juntamente com a distribuição de vida obtida na etapa anterior, aplica-se os procedimentos descritos na seção 4.2.5.3 para o estabelecimento do modelo de relacionamento estresse-envelhecimento acelerado. Um exemplo deste tipo de modelo é o relacionamento IPL-Lognormal (4.12), desenvolvido na seção supracitada. A figura 5.7 resume os principais elementos envolvidos neste processo.

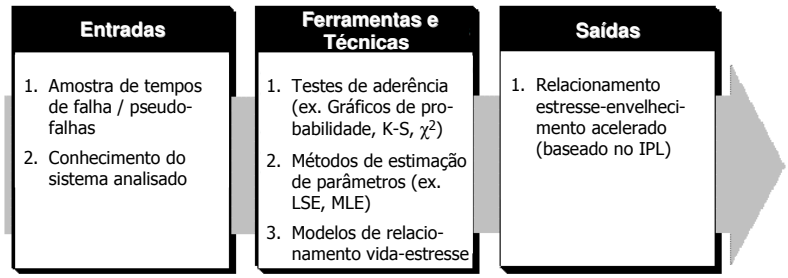

Figura 5.7: Principais elementos do 3º processo

5.6. ESTIMAÇÃO DA DISTRIBUIÇÃO DE VIDA DO NÍVEL DE USO

Com base no relacionamento estresse-envelhecimento acelerado, estimam-se os parâmetros do modelo a partir de um dos métodos de estimação utilizados na etapa anterior. Da mesma forma como no caso das distribuições de vida, é possível se estimar os parâmetros deste modelo tanto através de técnicas gráficas quanto analíticas.

De acordo com Nelson (2004, p. 168), para a distribuição Lognormal o método dos mínimos quadrados é o que proporciona estimativas mais acuradas, bem como limites de confiança exatos. Já para as distribuições Weibull e Exponencial, segundo o mesmo autor, o método recomendado é o da máxima verossimilhança. Deste modo, o método proposto adota estas considerações para a implementação desta etapa.

Após estimar os parâmetros da distribuição de vida, faz-se necessário estimar os parâmetros específicos do relacionamento IPL. Estes também podem ser estimados tanto de forma gráfica quanto analítica. A partir do enfoque gráfico, o método sugere os seguintes passos:

a) Criar um gráfico $\log_e$-$\log_e$ com o tempo de falha representado no eixo y e o estresse no eixo x, como apresentado na figura 5.8;
b) Para cada nível de estresse, marcar os pontos referentes aos valores estimados para o parâmetro de vida média. Para as distribuições consideradas neste trabalho, estes parâmetros estão listados no quadro 4.4;
c) Ajustar o modelo IPL linearizado aos pontos desenhados no item (b) e estimar os parâmetros como segue:

$$\hat{n} = \frac{\ln(T2) - \ln(T1)}{\ln(S2) - \ln(S1)},$$

onde T1 e T2 são os tempos de falha obtidos para os níveis de estresse S1 e S2 através do gráfico (figura 5.8). Posteriormente, resolve-se o IPL com respeito a *K*, como segue:

$$\tau(V) = \frac{1}{K \cdot V^n} = \hat{K} = \frac{1}{\tau \cdot V^{\hat{n}}},$$

onde:

V é o valor da variável de estresse (no eixo x da figura 5.8);

τ é o valor da vida média (eixo y da figura 5.8) que corresponde à *V*.

No passo (c), o valor de *V* pode ou não ser o mesmo do nível de uso, sendo uma escolha do experimentador. A figura 5.8 apresenta o gráfico usado para estes procedimentos.

Além do método gráfico, o experimentador tem a disposição métodos analíticos (ex. MLE), implementados amplamente pelos pacotes de software já citados.

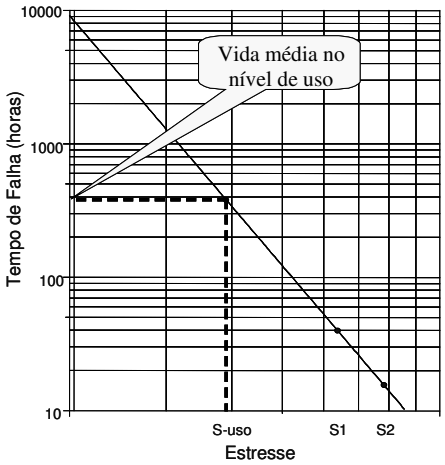

Figura 5.8: Gráfico na escala log-linear para estimar os parâmetros do modelo IPL

O relacionamento estresse-envelhecimento é um modelo de regressão linear simples. Portanto, a partir da estimação dos seus parâmetros deve-se aplicar os procedimentos (ex. análise dos resíduos) que são normalmente utilizados para validar a adequação deste tipo de modelo, os quais são procedentes da teoria de regressão linear. O capítulo 17 de Meeker e Escobar (1998) é dedicado à análise e diagnóstico de modelos de regressão para análise de dados de ensaios acelerados. Complementarmente, o capítulo 3 de Neter *et al.* (1996) discute detalhadamente os aspectos envolvidos no diagnóstico e tratamento deste tipo de modelo.

Confirmada a validade do modelo, as medidas de interesse no nível de uso podem ser estimadas com o relacionamento estresse-envelhecimento acelerado. A seção 4.2.5.3 apresentou alguns exemplos de medidas de confiabilidade (ex. taxa de falha) para o caso do relacionamento IPL-Lognormal. A figura 5.9 apresenta os elementos deste processo.

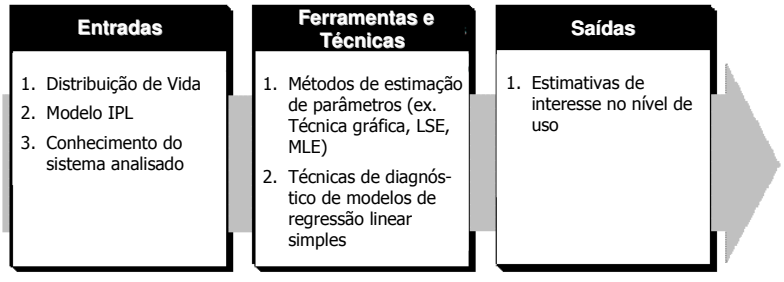

Figura 5.9: Principais elementos do 4º processo

CAPÍTULO 6

AVALIAÇÃO EXPERIMENTAL DO MÉTODO PROPOSTO

6.1. INTRODUÇÃO

Este capítulo apresenta a aplicação do método proposto na forma de um estudo experimental. Este estudo teve como principal objetivo avaliar as suposições admitidas, no capítulo 5, durante a especificação dos processos do método e a seleção das técnicas e ferramentas adotadas.

Primeiramente, será apresentado o software que será objeto da aplicação do método proposto, bem como as justificativas para a sua escolha. Posteriormente, serão descritos os principais elementos do ambiente de teste e a instrumentação utilizada. Na seqüência, a execução dos processos do método será apresentada, em conjunto com a análise dos seus resultados.

6.2. SOFTWARE ANALISADO

O software escolhido para ser estudado foi o servidor web *httpd*. O *httpd* é um dos projetos do Apache Group (www.apache.org), sendo mundialmente conhecido pelo nome Apache *web server* ou apenas Apache. As principais justificativas para a sua escolha foram:

- Comprovação do envelhecimento de software: Com base nos resultados reportados em Li, Vaidyanathan e Trivedi (2002), tem-se comprovado que o *httpd* possui faltas relacionadas ao envelhecimento que, ao serem ativadas, causam o vazamento de memória em seus processos. Estes resultados satisfazem um dos requisitos para a utilização do método proposto, que é a prévia comprovação da existência de envelhecimento de software no sistema sob teste.
- Popularidade: Segundo pesquisa[34] do Netcraft (2006a), atualmente o *httpd* responde por aproximadamente 61,25% do mercado mundial de servidores web. A figura 6.1 ilustra esta participação. Deste modo, entende-se que um estudo que contribua para a

[34] Realizada em junho de 2006 com 85.541.228 endereços de URL.

compreensão do problema do envelhecimento de software no *httpd* tem uma grande abrangência.

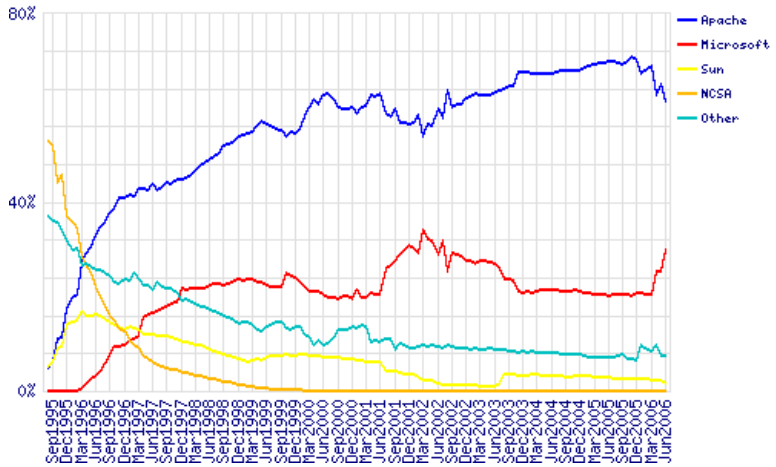

Figura 6.1: Participação de mercado por software servidor web (4 principais)
Fonte: Netcraft (2006a)

- Facilidade de replicação do experimento: O *httpd* é um software livre e, portanto, seu código fonte está disponível e pode ser utilizado sem restrições. Esta característica permite com que outros pesquisadores possam ter acesso ao mesmo programa avaliado neste trabalho, o que facilita a replicação dos procedimentos realizados nesta etapa experimental. Como exemplo, o ambiente de teste implementado para esta fase da pesquisa possui as mesmas funcionalidades do ambiente descrito em Li, Vaidyanathan e Trivedi (2002), o que permitiu realizar algumas comparações dos resultados obtidos em ambos os trabalhos.
- Importância para a indústria: Por ser considerado o servidor web mais usado atualmente, o *httpd* tem sido integrado em diversos produtos comerciais. O quadro 6.1 apresenta alguns exemplos de produtos comerciais baseados no *httpd*, seja na forma de customizações ou com a sua integração a outros componentes de software proprietários. Além de fazer parte de diversos produtos, o *httpd* tem sido extensivamente usado por portais web de grande porte. Alguns exemplos estão listados no quadro 6.2.

Produto	Software Servidor Web	Fabricante
Oracle HTTP server	Apache modificado	Oracle
Secure Web Server for OpenVMS	Apache	Hewlett-Packard (HP)
Apache BS2000/OS	Apache	Fujitsu-Siemens
IBM HTTP Server	Apache modificado	IBM

Quadro 6.1: Produtos comerciais baseados no *httpd*

Portal/Site	Software Servidor Web
www.google.com	GWS (Apache modificado)
www.orkut.com	GFE (Apache modificado)
www.cnn.com	Apache
www.bb.com.br	IBM HTTP Server/V5R3M0 (Apache modificado)
www.uol.com.br	Apache
www.terra.com.br	Apache

Quadro 6.2: Portais web que utilizam o *httpd* em sua infra-estrutura
Fonte: Netcraft (2006b)

Com base nas justificativas descritas nesta seção, o software *httpd* foi selecionado para esta etapa da pesquisa. A próxima seção apresenta o ambiente de teste utilizado.

6.3. AMBIENTE DE TESTE

O instrumental de hardware utilizado nesta etapa foi baseado em dois computadores, sendo um equipamento no papel de gerador de tráfego (CPU: Celeron 1.2 Ghz, RAM: 512 megabytes, NIC Ethernet: 100 Mbps) e o outro para executar o servidor web (CPU: P4 2.8 Ghz, RAM: 512 megabytes, NIC Ethernet: 100 Mbps). A interconexão dos computadores foi realizada através de um *switch* Ethernet de 100 Mbps. A figura 6.2 ilustra este ambiente.

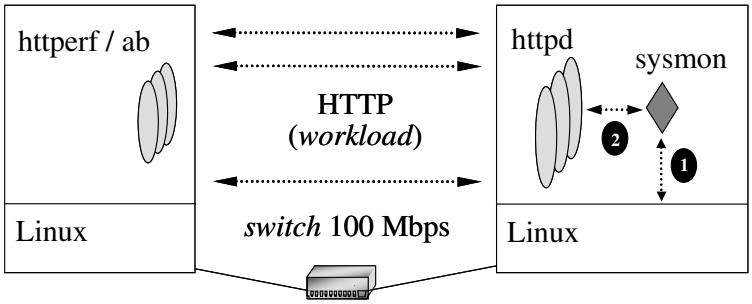

Figura 6.2: Bancada de testes

O sistema operacional utilizado foi o Red Hat Enterprise Linux AS (*kernel* Linux 2.4.21-4). O software usado para monitorar o ambiente servidor foi implementado para o propósito deste trabalho e recebeu o nome de *sysmon*. Como se pode observar na figura 6.2, o *sysmon* realiza dois tipos básicos de monitoração. A primeira monitora os recursos do sistema operacional, sendo realizada através da leitura e interpretação de diversos arquivos no diretório /proc (ex. /proc/meminfo) do Linux. No segundo tipo de monitoração, o *sysmon* coleta dados específicos sobre os processos *httpd*, como por exemplo, a quantidade de memória utilizada por processo. Esta coleta também é baseada em arquivos do diretório /proc.

Para realizar a geração das cargas de trabalho, foram utilizados os programas *httperf* (MOSBERGER; JIN, 1998) e *ab*, este último distribuído junto com o *httpd*. O *ab* foi usado de forma complementar ao *httperf*, para os casos onde o número de conexões concorrentes permanece constante, já que esta funcionalidade não está disponível na interface padrão do *httperf*. Também, o *ab* serviu para validar os resultados obtidos com o *httperf*.

O software servidor web utilizado foi o *httpd* (versão 2.0.46) compilado como MPM *prefork* (THE APACHE GROUP, 2005). O número máximo de 300 processos *httpd* foi usado durante os testes, resultando em uma capacidade de atendimento de até 300 sessões HTTP simultâneas. A configuração adotada criou os 300 processos *httpd* no início de cada teste, mantendo-os carregados na memória e disponíveis para execução até o fim dos testes. Deste modo, evitou-se a remoção dos processos ociosos como prevê o comportamento padrão do *httpd* (THE APACHE GROUP, 2005). De forma geral, esta abordagem objetivou a maior exposição dos processos aos efeitos do envelhecimento, garantindo que cada processo executasse durante todo o período de teste. A vantagem deste enfoque, em relação ao adotado em Li, Vaidyanathan e Trivedi (2002), refere-se à maximização do tempo de exposição dos processos aos efeitos do envelhecimento, contribuindo na redução do tempo para se detectar a manifestação destes efeitos sobre os processos monitorados. O quadro 6.3 apresenta a customização realizada no arquivo *httpd.conf* para implementar o comportamento descrito anteriormente. Com exceção destas diretivas, as demais configurações seguiram o padrão (*default*) de instalação do *httpd*.

Diretiva (http.conf)	Valor
MaxKeepAliveRequests	0
ServerLimit	300
StartServers	300
MinSpareServers	300
MaxSpareServers	0
MaxClients	300
MaxRequestsPerChild	0

Quadro 6.3: Configuração do *httpd* para os experimentos

Para o tratamento dos dados, foram utilizados os softwares STATISTICA (www.statsoft.com), @Risk Best Fit (www.palisade.com), Weibull++ 7 (www.reliasoft.com) e ALTA Pro 6 (www.reliasoft.com). Os dois últimos fazem parte do pacote de licenças disponibilizadas pela Reliasoft para a UFSC, sob os termos do contrato de cooperação entre as duas entidades.

A seguir serão apresentados os resultados obtidos com a execução de cada etapa do método.

6.4. APLICAÇÃO DO MÉTODO PROPOSTO

Como detalhado no capítulo 5 (ver figura 5.1), os processos definidos pelo método são: (i) seleção do fator de envelhecimento, (ii) planejamento e execução do TEA, (iii) modelagem do relacionamento estresse-envelhecimento acelerado e (iv) estimação da distribuição de vida para o nível de uso. O restante deste capítulo apresenta os resultados da execução de cada uma destas etapas.

6.4.1. Seleção do fator de envelhecimento

Como descrito na seção 5.3, este processo é dividido em duas partes:

a) Teste experimental dos fatores candidatos;
b) Análise dos efeitos dos fatores sobre o envelhecimento.

A seguir os resultados destas etapas.

6.4.1.1. Teste experimental dos fatores candidatos

Os fatores candidatos foram escolhidos com base nos resultados reportados em Li, Vaidyanathan e Trivedi (2002) e Matias Jr. *et al*. (2004). São eles:

- Tamanho médio de página (TMP);
- Tipo de página (TP);
- Taxa de requisições HTTP (TR).

Como definido no método proposto, o DOE segue um projeto fatorial $2^k r$, o que significa que cada fator deve estar associado a valores em dois níveis. Neste trabalho estes níveis são denominados de *normal* e *alto*. Os valores atribuídos ao nível *normal* foram aqueles que, a combinação dos três fatores, exerceu uma carga de aproximadamente 50% da capacidade do servidor web. No nível *alto* a capacidade exercida foi de aproximadamente 90%.

Para a definição do valor associado ao nível *normal* do fator TMP, foi realizada uma pesquisa com 6.000 páginas HTML de um provedor de serviços Internet (ISP). Foram

selecionadas as dez páginas mais acessadas em um período de três meses, obtendo um TMP igual a 196 *kilobytes*. Quanto ao nível *alto* deste fator, buscou-se um tamanho representativo daqueles casos em que o servidor web transfere objetos de dados maiores do que páginas HTML convencionais. Atualmente, é uma prática comum a distribuição de arquivos binários (ex. atualizações de software) através de servidores web, em alternativa aos repositórios FTP. A fim de representar estes casos, trabalhou-se com uma amostra com três tamanhos de arquivos de atualização de antivírus. O valor médio desta amostra foi de 2 *megabytes*, o qual foi associado ao nível alto do fator TMP.

O fator TP se refere à forma como o conteúdo da URL requisitada está disponível, ou seja, na forma de conteúdo estático ou conteúdo criado dinamicamente. O nível *normal* deste fator representou objetos com conteúdo estático, tais como arquivos HTML, imagens (ex. jpg), dentre outros. Já o nível *alto* representou conteúdo criado dinamicamente, como por exemplo, a partir de programas CGI (*Common Gateway Interface*). Para implementar o nível *alto* deste fator foi criado um programa CGI, em linguagem C, para gerar páginas HTML de acordo com os tamanhos definidos pelo fator TMP.

Para determinar os níveis do fator TR, foram realizados testes de carga para encontrar a capacidade máxima do servidor em cada configuração de TMP e TP para seus dois níveis (*normal* e *alto*) previamente definidos. Depois de identificada a capacidade máxima, calculou-se os percentuais 50% e 90% deste valor, o que correspondeu, respectivamente, aos níveis *normal* e *alto* do fator TR. As figuras 6.3 e 6.4 apresentam os resultados do teste de carga para TMP e TP no nível *normal*, utilizando tanto o *httperf* quanto *ab*.

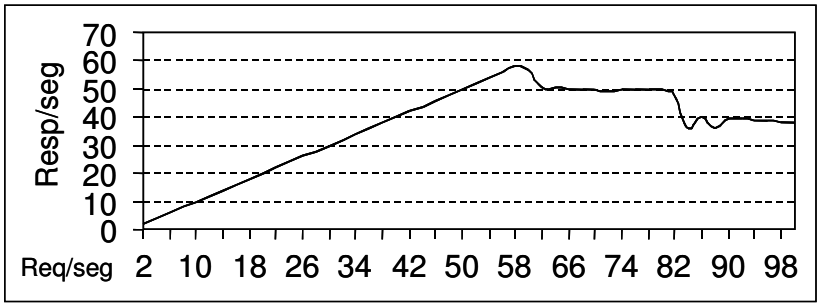

Figura 6.3: Teste de carga com *httperf* (TMP=196 KB; TP=estático)

A vazão (*throughput*) máxima, em termos de atendimento de conexões pelo servidor, para a configuração selecionada, foi de 59,4 (*httperf*) e 59,8 (*ab*) respostas por segundo. Ressalta-se que nos testes de carga utilizando o *httperf*, a carga de trabalho foi implementada

incrementando o número de requisições por segundo (valores do eixo x). Já nos testes com o *ab* incrementou-se o número de conexões concorrentes.

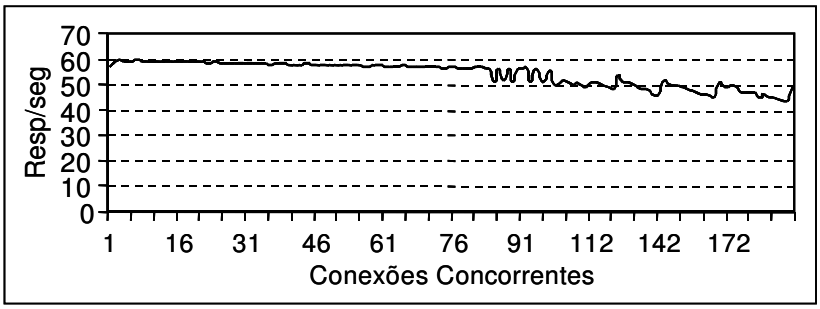

Figura 6.4: Teste de carga com *ab* (TMP=196 KB; TP=estático)

A tabela 6.1 apresenta todos os valores do fator TR obtidos com os testes de carga. Estes valores foram obtidos com a ferramenta *httperf*.

Tabela 6.1: Níveis do fator taxa de requisições (TR)

Configuração do teste de carga	TR (req/seg)	
	Normal (50%)	Alto (90%)
TMP=196 kilobytes; TP=estático	29,7	53,4
TMP=196 kilobytes; TP=dinâmico	2,9	5,2
TMP=2 megabytes; TP=estático	19,0	34,2
TMP=2 megabytes; TP=dinâmico	2,0	3,6

Em Li, Vaidyanathan e Trivedi (2002) e Matias Jr. *et al.* (2004), foi verificado experimentalmente que os efeitos do envelhecimento de software sobre o *httpd* tem como conseqüência à degradação da memória virtual do sistema servidor web. As evidências apresentadas por ambos os trabalhos apontam como causa desta degradação o vazamento de memória (*memory leaks*) no *httpd*, causando o aumento progressivo no tamanho dos seus processos. Neste caso, definiu-se como variável resposta (y) do DOE o incremento no tamanho dos processos *httpd*. Assim sendo, o *sysmon* foi configurado para monitorar o tamanho individual de cada processo *httpd*, reportando este valor a cada 100 requisições processadas pelo servidor.

A partir da definição dos fatores e seus níveis, juntamente com a variável resposta, o método prevê a realização de um experimento piloto, cujo resultado será usado para estimar o número de replicações (tamanho da amostra) necessário para o projeto experimental. Como definido pelo método, o experimento piloto é realizado com valores médios associados aos níveis dos fatores. Neste estudo, os valores médios adotados estão apresentados a seguir:

- Número de replicações: 20;
- Número de requisições por replicação: 5.000;
- TMP (bytes): 1.148.928 bytes;
- TP: dinâmico;
- TR (req/seg): 4,9 (70% da capacidade total).

O valor de TP foi definido com o nível *alto*, pois nos testes de carga verificou-se uma maior variabilidade nos resultados neste nível. O valor de TR considerou 70% (média entre os níveis *normal* e *alto*) da capacidade total do servidor para a configuração dos demais fatores do experimento piloto. O número de 5.000 requisições foi suficiente para avaliar os efeitos do envelhecimento sobre a variável resposta, tendo sido definido a partir dos resultados observados durante os testes de carga. A tabela 6.2 apresenta o resultado do experimento piloto. Para cada replicação, tem-se o incremento total no tamanho dos processos *httpd* causado pelo envelhecimento de software durante o teste.

Tabela 6.2: Amostra piloto para o cálculo do número de replicações do DOE

Replicação	*y*	Replicação	*y*
1	71308	11	63188
2	65232	12	66560
3	64652	13	68556
4	69304	14	62744
5	66640	15	65356
6	68288	16	66792
7	69440	17	64832
8	71596	18	64124
9	63704	19	62224
10	60932	20	60900

Com base nesta amostra, a equação 5.5 foi resolvida para obter o número de replicações do DOE, o que resultou em 21 replicações como descrito a seguir:

$$n = \frac{s_0^{\,2}}{E_0^2 \cdot 2^{k-2}} = \frac{10112583{,}20}{500^2 \cdot 2^{3-2}} = 20{,}22517 \approx 21,$$

onde E_0 foi definido como 500 bytes.

Deste modo, o DOE foi executado com 21 replicações para cada um dos oito tratamentos. A tabela 6.3 apresenta os tratamentos e seus respectivos resultados, em termos do valor médio das replicações para a variável resposta. Os valores da variável resposta para cada replicação estão apresentados na tabela B.1 do apêndice B.

Tabela 6.3: Resultado do DOE

Tratamento	I	A	B	C	AB	AC	BC	ABC	$\overline{y}$
T1	1	-1	-1	-1	1	1	1	-1	24000
T2	1	1	-1	-1	-1	-1	1	1	24000
T3	1	-1	1	-1	-1	1	-1	1	24000
T4	1	1	1	-1	1	-1	-1	-1	24000
T5	1	-1	-1	1	1	-1	-1	1	49942
T6	1	1	-1	1	-1	1	-1	-1	51505
T7	1	-1	1	1	-1	-1	1	-1	68444
T8	1	1	1	1	1	1	1	1	67771
		A=TR		B=TMP		C=TP			

A figura 6.5 apresenta o comportamento (média das replicações) do incremento no tamanho dos processos *httpd* durante a execução de cada tratamento.

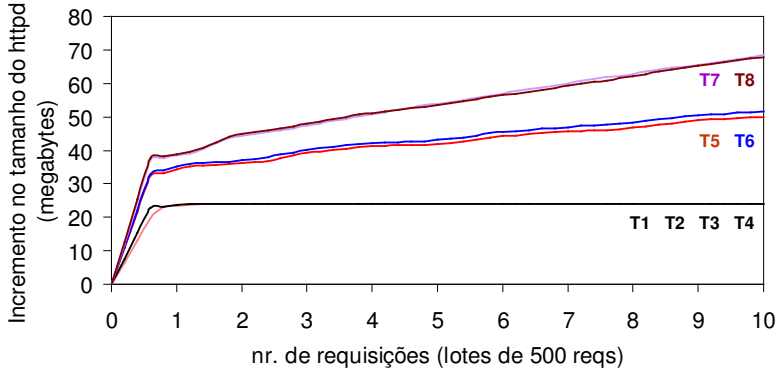

Figura 6.5: Efeito do envelhecimento sobre o tamanho dos processos *httpd*

Como se pode observar, em todos os oito tratamentos o tamanho dos processos passou por um período de incremento inicial durante as primeiras 500 requisições, o qual será chamado de período de inicialização. Após este período, nos quatro primeiros tratamentos não ocorreu o envelhecimento dos processos *httpd*, haja vista que o tamanho total dos processos se manteve em 24 megabytes durante todo o teste após o período de inicialização. Já para os demais tratamentos (T5 - T8), houve um incremento monotônico no tamanho dos processos após o período de inicialização, prolongando-se até o final dos testes. Em T5 e T6 este incremento correspondeu, respectivamente, à aproximadamente 17 e 18 *megabytes*. O comportamento do envelhecimento em T7 e T8 foi similar ao observado nos dois tratamentos precedentes, diferenciando-se apenas na intensidade, que foi de aproximadamente 31 e 30 *megabytes*, respectivamente. Estes resultados indicaram que para os quatro primeiros tratamentos, os processos *httpd* não sofreram de envelhecimento de software, ou seja, não houve o vazamento de memória nestes quatro cenários. Já para os quatro últimos tratamentos,

ficou claro a existência dos efeitos do envelhecimento no aumento do tamanho dos processos, causando a degradação progressiva da memória no sistema operacional do servidor web. A seguir serão apresentados os resultados da estimação dos efeitos de cada fator sobre a variável resposta, a fim de identificar quais fatores mais contribuíram para o envelhecimento dos processos *httpd*.

6.4.1.2. Análise dos efeitos dos fatores

Os resultados da etapa anterior apresentaram evidências de que o padrão de ativação das faltas relacionadas ao envelhecimento esteve presente apenas nos quatro últimos tratamentos. A fim de verificar quais foram os fatores que fizeram parte deste padrão, o método proposto define a análise quantitativa da contribuição de cada fator para o envelhecimento do sistema sob teste. Para isso, o método prevê a realização de uma ANOVA em conjunto com a resolução da matriz de sinais do DOE. Ambos procedimentos foram apresentados na seção 5.3.2.

O quadro 6.4 apresenta os resultados da ANOVA, juntamente com o teste *F* realizado com nível de significância de 5% (α=0,05). Vale ressaltar, que devido aos quatro primeiros tratamentos não terem apresentado variabilidade nos dados (ver tabela 6.3), a realização do teste *F* neste caso não é adequada. Contudo, o mesmo foi realizado apenas para ilustrar a sua aplicação durante a implementação do método proposto.

Fonte de variação	Soma dos quadrados	gl	Quadrados médios	Razão *f*
A	2,E+06	1	2,08E+06	0,49
B	3,E+09	1	3,17E+09	745,69
C	5,E+10	1	5,27E+10	12380,00
A*B	1,E+07	1	1,31E+07	3,08
A*C	2,E+06	1	2,08E+06	0,49
B*C	3,E+09	1	3,17E+09	745,69
A*B*C	1,E+07	1	1,31E+07	3,08
Erro	7,E+08	160	4,26E+06	

Quadro 6.4: Resultado da ANOVA

Se o teste *F* fosse considerado válido, a interpretação da última coluna do quadro seria no sentido de que os efeitos significativos foram provenientes dos fatores B e C, como também da interação BC, haja vista que o valor do *F*-crítico (F_c) tabelado é 3,9, para α=0,05, numerador=1 e denominador=160.

No quadro 6.5, são apresentados os resultados da resolução da matriz de sinais do DOE descrito anteriormente. A segunda linha apresenta o efeito estimado dos fatores, calculado a partir da equação 5.3. A quarta linha traz os resultados das somas dos quadrados: totais (ver equação 5.5) e dos fatores e suas interações (ver quadro 5.2). A última linha do quadro apresenta a variação estimada de *y*, explicada pelos fatores e suas interações, obtida a partir da equação 5.7.

I	A	B	C	AB	AC	BC	ABC
41707,86	111,24	4345,95	17707,86	-279,43	111,24	4345,95	-279,43
SQT	**SQA**	**SQB**	**SQC**	**SQAB**	**SQAC**	**SQBC**	**SQABC**
2,81E+09	9,90E+04	1,51E+08	2,51E+09	6,25E+05	9,90E+04	1,51E+08	6,25E+05
Estimação da contribuição dos fatores e interações sobre a variação de y							
	0,00%	5,37%	89,20%	0,02%	0,00%	5,37%	0,02%

Quadro 6.5: Matriz de sinais resolvida para o projeto experimental realizado

Com base nos resultados obtidos com a computação da matriz de sinais, verificou-se que o fator de maior influência no envelhecimento foi o tipo de página (C), seguido do tamanho de página (B). O fator taxa de requisição (A) não apresentou influência significante neste estudo. Estes resultados corroboram com aqueles obtidos com a ANOVA e, com base nestas estimativas, o fator de envelhecimento (F_E) será a composição dos fatores TMP e TP que, individualmente e combinados, representaram 99,94% da variação de *y* no experimento realizado.

6.4.2. Planejamento e execução do TEA

A implementação deste processo exigiu a definição dos elementos envolvidos com a forma do teste e o plano experimental, como descrito nas seções 5.4.1 e 5.4.2.

6.4.2.1. Elementos de forma do TEA

Foi adotada como medida de performance a própria variável resposta do DOE, como sugere o método proposto. Para cada ensaio do TEA foram processadas 50.000 requisições HTTP, sendo que a cada 100 requisições foi realizada uma inspeção no tamanho dos processos *httpd*. Desta forma, cada caminho de degradação foi composto de 500 valores de medições do tamanho total dos processos *httpd*.

No papel de variável de estresse o método utiliza o conceito de F_E, que de acordo com a seleção realizada na seção anterior, será composto dos fatores TP e TMP. O fator TP é do tipo qualitativo e suporta apenas dois valores (estático e dinâmico), não permitindo a implementação de três níveis de estresse como sugerido pelo método. Neste caso, a carga de trabalho usada no TEA considerou o fator TP no nível dinâmico e o fator TMP assumindo três valores que corresponderam aos três níveis de estresse propostos para o plano experimental do TEA (ver seção 5.4.2). Esta definição levou em consideração os resultados do DOE, especialmente dos tratamentos T5, T6, T7 e T8. Como observado na tabela 6.3, os tratamentos T7 e T8 resultaram em um incremento no tamanho dos processos de aproximadamente 1,3 vezes o valor obtido com os tratamentos T5 e T6. Esta diferença foi causada pela variação do fator TMP, já que nestes quatro experimentos os demais fatores (TP e TR) permaneceram inalterados. Deste modo, os três níveis de estresse foram implementados atribuindo três diferentes valores ao fator TMP, que corresponderam a duas, três e quatro vezes o valor do tamanho médio de páginas caracterizado para o nível de uso do sistema analisado.

Com relação às condições de teste, a configuração da carga de trabalho usada no TEA adotou uma taxa de requisições maximizada, assumindo valores que corresponderam a 90% da capacidade de atendimento do sistema para cada nível de estresse. Isso foi possível, pois os resultados da seção anterior demonstraram que a taxa de requisição (TR) não exerceu influência sobre o envelhecimento dos processos, o que permitiu incrementar a taxa de requisições sem influenciar no envelhecimento, contribuindo para a redução do tempo de experimentação.

Com respeito à forma de aplicação da carga de estresse, o método define a utilização de estresse constante. Deste modo, a carga de trabalho foi implementada gerando tráfego sintético com a ferramenta *httperf*, configurada para manter a taxa de requisições constante para o mesmo tamanho e tipo de página.

Para os mecanismos de censura do TEA, foi definido como limiar de falha (D_f) o valor de 400 megabytes e como duração do teste (D_t) a quantidade de 50.000 requisições. Portanto, cada ensaio finalizou quando o incremento no tamanho dos processos *httpd* atingiu ou ultrapassou o valor de D_f ou quando o número de requisições atendidas pelo servidor fosse igual à D_t. Este valor de D_f foi especificado, verificando que após a carga do sistema operacional e dos 300 processos *httpd*, a disponibilidade de memória principal foi de aproximadamente 410 megabytes. Em testes preliminares de envelhecimento acelerado, observou-se que quando o tamanho total dos processos *httpd* aproximou-se deste valor (D_f), o Linux começou a salvar páginas da memória principal para o disco (*swapping*), devido à

baixa disponibilidade de memória causada pelo crescimento dos processos *httpd*. Este comportamento não é desejável para um sistema servidor web, haja vista o incremento no seu tempo de resposta provocado pelo excesso de operações de E/S durante o *swapping*, o que muitas vezes causa o encerramento de conexões por *timeout*. Este problema é abordado em detalhes em Matias Jr. e Freitas Filho (2006). Com respeito ao valor de D_t, este foi definido com o objetivo de ter no mínimo um dos níveis de estresse com seus caminhos de degradação atingindo o limiar D_f. Uma síntese das definições desta seção está apresentada na tabela 6.4.

Tabela 6.4: Sumário dos elementos de forma do TEA

	Nível de uso (NU)	Sl (2 x NU)	S2 (3 x NU)	S3 (4 x NU)
Tamanho de página (KB)	196	392	588	784
Taxa de requisições do F_E (req/sec)	-	29	19	14
Número de requisições (D_t)	-	50.000	50.000	50.000
Número de inspeções (t)	-	500	500	500
Limiar de falha (D_f) (megabytes)	-	400	400	400

6.4.2.2. Plano experimental

O plano experimental adotado foi um plano tradicional, baseado nos três níveis de estresse definidos na seção anterior. Para o cálculo do tamanho da amostra (número de replicações) do TEA, realizou-se o procedimento descrito no apêndice A, o qual requer uma amostra piloto com 21 replicações no caso do plano tradicional escolhido (7 replicações em cada nível de estresse). A amostra piloto de tempos de falha (TTF) e *pseudofalha* (TTPF) se encontra no quadro 6.6.

TTPF (S1)	TTPF (S2)	TTF (S3)
9501	1421	405
9620	1459	407
9982	1475	416
10332	1517	422
11180	1522	427
12593	1552	434
12622	1553	439

Quadro 6.6: Amostra piloto do TEA

A figura 6.6 apresenta os caminhos de degradação obtidos com a execução do TEA piloto, os quais foram usados para obter os valores do quadro 6.6.

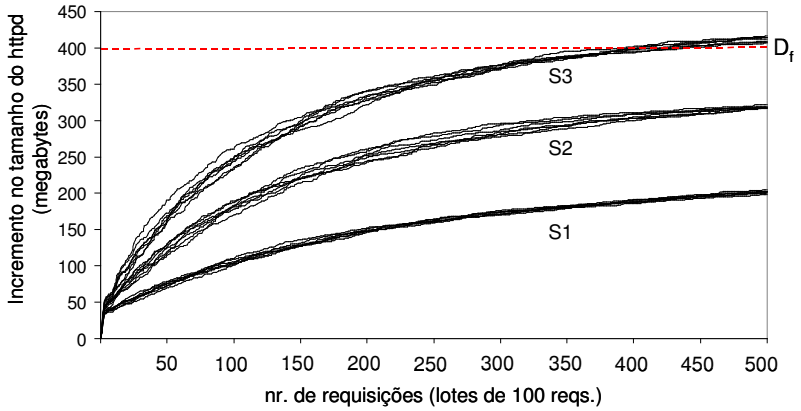

Figura 6.6: Caminhos de degradação do TEA piloto

Para os dois primeiros níveis de estresse (S1 e S2), os 14 caminhos de degradação não alcançaram o limiar D_f no período de duração dos ensaios (D_t=500). Para estes casos, foi necessário ajustar um modelo de regressão para cada caminho de degradação, a fim de se obter um tempo de *pseudofalha* (ver seção 5.4.3). Tanto para S1 quanto para S2, o melhor ajuste foi de um modelo logarítmico. Os modelos considerados nesta etapa estão listados no quadro 6.7.

Modelo	Equação
Logarítmico	y=a*ln(x)+b
Potência	y=b*(x^a)
Lloyd-Lipow	y=a-b/x
Linear	y=a*x+b
Exponencial	y=b*exp(a*x)

Quadro 6.7: Modelos de regressão testados

Os valores dos parâmetros dos modelos logarítmicos, para cada caminho de degradação, estão listados na tabela B.2 (ver apêndice B). Estes valores foram estimados pelo método dos mínimos quadrados (LSE) utilizando o software Weibull++ 7. No apêndice B, também são apresentadas as figuras (B.1 e B.2) que ilustram os caminhos de degradação para S1 e S2, com os seus respectivos modelos logarítmicos ajustados. Nestas figuras, os dados observados estão representados por círculos e os modelos teóricos por linhas, destacando que os ajustes dos modelos de regressão foram realizados com a metade da amostra, considerando os valores a partir do ponto 250 (do eixo x). A decisão por se trabalhar com a segunda metade da amostra objetivou não utilizar os dados do período de inicialização, abordagem esta que apresentou os melhores resultados em termos da qualidade do ajuste (R^2) dos modelos.

Para o terceiro nível de estresse (S3), a duração do TEA foi suficiente para que os caminhos de degradação alcançassem D_f, observando-se diretamente os tempos de falha (TTF) e não sendo necessário realizar extrapolações dos caminhos de degradação para a obtenção de TTPFs.

O próximo passo foi verificar qual das distribuições de probabilidades, consideradas no algoritmo do cálculo da amostra (ver item "I.c" da seção A.2.1), melhor se ajustava aos dados da amostra piloto. Para isso foram realizados testes de aderência utilizando o módulo *distribution wizard* do software Weibull++ 7. Os critérios usados para a composição do *ranking* de melhor ajuste foram o valor da função *log-verossimilhança* (*Lk*) e o coeficiente de correlação linear de *Pearson* (ρ), cujos métodos de estimação de parâmetros foram, respectivamente, MLE e LSE. Os testes K-S e Qui-quadrado não foram utilizados devido ao tamanho reduzido da amostra. O resultado do teste de aderência está apresentado na tabela 6.5.

Tabela 6.5: Resultado do teste de aderência com a amostra piloto do TEA

Nível de Estresse	Modelo	GOF		Ranking
		Lk	ρ (%)	
S1	Lognormal	59,60285	94,8340	1º
	Weibull	-60,12718	91,3959	2º
	Exponencial	-72,03237	-67,9887	3º
S2	Weibull	-36,43342	98,3961	1º
	Lognormal	-36,82059	96,9112	2º
	Exponencial	-58,19187	-76,8152	3º
S3	Lognormal	-27,35933	98,6593	1º
	Weibull	-27,45844	97,3205	2º
	Exponencial	-49,30555	-96,0616	3º

A distribuição Lognormal foi a que melhor se ajustou aos dados dos níveis de estresse S1 e S3. Para o nível S2, o melhor ajuste ocorreu com a distribuição Weibull seguida da Lognormal. A diferença entre estas duas distribuições, no nível S2, foi pequena e como a Lognormal demonstrou o melhor ajuste nos demais níveis, decidiu-se adotá-la também para S2. No apêndice B, as figuras B.3, B.4 e B.5 apresentam os gráficos de probabilidade Lognormal para S1, S2 e S3, demonstrando a boa aderência desta distribuição para os três níveis de estresse.

A próxima etapa do algoritmo foi transformar os dados de acordo com as regras definidas no item "I.d" da seção A.2.1 (ver apêndice A). Como definido pelo algoritmo, se o melhor ajuste for de uma distribuição de probabilidades Lognormal, a transformação deve ser

o Log_{10} dos dados observados. Portanto, o resultado da transformação dos valores observados (quadro 6.6) está listado no quadro 6.8.

TTPF (S1)	TTPF (S2)	TTF (S3)
3,9778	3,1526	2,6075
3,9832	3,1641	2,6096
3,9992	3,1688	2,6191
4,0142	3,1810	2,6253
4,0484	3,1824	2,6304
4,1001	3,1909	2,6375
4,1011	3,1912	2,6425

Quadro 6.8: Resultado do Log_{10} da amostra piloto do TEA

A partir dos dados transformados, o número de replicações do TEA (n_{TEA}) foi calculado executando os procedimentos descritos no item II da seção A.2.1, os quais estão sumarizados a seguir:

(A.1) $\bar{x} = 2{,}752326$

(A.2) $\bar{y}_1 = 4{,}0320$, $\bar{y}_2 = 3{,}1758$, $\bar{y}_3 = 2{,}6245$

(A.3) $s_j = 0{,}0523$, $s_j = 0{,}0145$, $s_j = 0{,}0133$

(A.4) $s = 0{,}0323$

(A.5) $$n_{TEA} = \left\{1 + (x_0 - \bar{x})^2 \left[\frac{np}{\sum (x - \bar{x})^2}\right]\right\}\left(\frac{z_{\alpha/2}\sigma}{\zeta}\right)^2 = 34{,}73149,$$

onde:

$x_0 = \log_{10}(196) = 2{,}2923$

$\alpha = 0{,}95$, $z_{\alpha/2} = 1{,}96$

$r = (1 + m)$, m=0,10

$\zeta = \log_{10}(r) = \log_{10}(1{,}10)$

$\sigma \approx s$

Como resultado, o n_{TEA} calculado foi de 34,74. Tendo em vista a adoção de um plano tradicional, cuja alocação é idêntica nos três níveis de estresse, este valor foi arredondado para 36, resultando em 12 replicações para cada nível. Portanto, foram necessárias mais 15 replicações, adicionais às 21 já realizadas durante o TEA piloto. Deste modo, foram realizados mais cinco testes de envelhecimento acelerado, em cada nível de estresse,

resultando nos valores listados no quadro 6.9. A tabela 6.6 apresenta um sumário do plano experimental utilizado durante a execução do TEA.

TTPF (S1)	TTPF (S2)	TTF (S3)
9928	1353	382
10085	1379	394
11454	1631	398
12173	1749	448
13765	1760	450

Quadro 6.9: Amostra complementar para o TEA

Tabela 6.6: Plano tradicional adotado para a execução do TEA

Nível de estresse		Alocação	
Condição	TMP (F_E) (*kilobytes*)	Proporção π_i	Número de testes n_i
Uso	196	-	-
N_b	392	1/3	12
N_i	588	1/3	12
N_a	784	1/3	12

6.4.3. Modelagem do relacionamento estresse-envelhecimento acelerado

Primeiramente, foi selecionada a distribuição de probabilidades dos tempos de falha/*pseudofalha* para ser usada com o modelo de relacionamento estresse-envelhecimento acelerado (relacionamento vida-estresse). Os mesmos procedimentos realizados na seção anterior, para o teste de aderência com a amostra piloto do TEA, foram realizados com a amostra completa (36 valores). A tabela 6.7 apresenta os resultados obtidos. Como se pode verificar, a distribuição Lognormal, como no caso da amostra piloto, também demonstrou o melhor ajuste para os dados da amostra completa.

Tabela 6.7: Resultados do teste de aderência com a amostra completa do TEA

Nível de Estresse	Modelo	GOF		Ranking
		Lk	ρ (%)	
S1	Lognormal	-103,2170	96,5586	1º
	Weibull	-104,2556	92,1861	2º
	Exponencial	-123,7795	-71,3103	3º
S2	Lognormal	-74,7216	97,9346	1º
	Weibull	-75,9450	94,7224	2º
	Exponencial	-100,0034	-75,1620	3º
S3	Lognormal	-53,5946	99,1169	1º
	Weibull	-53,7244	98,3337	2º
	Exponencial	-84,4401	-79,2346	3º

Complementarmente à seleção da distribuição de probabilidades, verificou-se a suposição de mesmo parâmetro de escala (σ) para todos os níveis de estresse (ver seção 4.2.5.2). Para tanto, foram calculadas as estimativas dos parâmetros do modelo Lognormal, juntamente com seus intervalos de 90% de confiança – IC (90%). Estes resultados estão dispostos na tabela 6.8.

Tabela 6.8: Parâmetros da Lognormal estimados para os três níveis de estresse

Estresse	Parâmetros	Estimativa (MLE)	IC (90%)	
			LI	LS
S1	μ_1	9,3078	9,2487	9,3669
	σ_1	0,1245	0,0869	0,1782
S2	μ_2	7,3304	7,2907	7,3701
	σ_2	0,0837	0,0584	0,1198
S3	μ_3	6,0354	6,0105	6,0604
	σ_3	0,0525	0,0367	0,0752

Nas linhas sombreadas da tabela, verifica-se que existe interseção dos ICs estimados para o parâmetro de escala nos níveis S1 e S2, mas não para S3. Como explanado na seção anterior, durante os testes de envelhecimento acelerado para S3, observou-se a ocorrência de *swapping* na medida que os caminhos de degradação se localizavam próximos do valor pré-definido para D_f. Este comportamento altera o tamanho dos processos, influenciando nos valores de TTF observados em S3. Portanto, considerando os efeitos do *swapping* sobre os TTFs obtidos em S3, bem como a proximidade do limite superior (LS) do IC obtido em S3 com os valores do limite inferior (LI) dos ICs estimados para S1 e S2, optou-se pelo ajuste de $\hat{\sigma}$ para um valor que esteja dentro ou próximo do IC de cada nível. Com base nos valores da tabela 6.8, este ajuste poderia ser para um valor próximo de 0,08. Conduzindo esta mesma análise com o software ALTA Pro 6, o valor de $\hat{\sigma}$ foi automaticamente ajustado para 0,09 em todos os níveis. Neste sentido, adotou-se $\hat{\sigma} = 0{,}09$, pois o ALTA Pro também foi utilizado nas próximas etapas, especificamente para o ajuste do modelo de relacionamento estresse-envelhecimento acelerado e na estimação de medidas de confiabilidade para o nível de uso.

No capítulo 5 (ver seção 5.5.2), o método proposto preconiza a utilização do relacionamento IPL juntamente com a distribuição de vida selecionada para os três níveis de estresse, a fim de estabelecer a relação estresse-envelhecimento acelerado. Portanto, o modelo de relacionamento obtido nesta etapa foi o IPL-Lognormal.

6.4.4. Estimação da distribuição de vida para o nível de uso

Nesta seção, o modelo de relacionamento IPL-Lognormal foi usado para estimar a distribuição dos tempos de falha (distribuição de vida) para o nível de uso do servidor web analisado. Inicialmente, foram estimados os parâmetros do modelo IPL-Lognormal com o auxílio do software ALTA Pro 6. Estas estimativas estão apresentadas na tabela 6.9 e foram realizadas com base na amostra de 36 valores obtida com a execução do TEA. Maiores detalhes sobre cada parâmetro podem ser obtidos no exemplo de utilização do IPL-Lognormal descrito na seção 4.2.5.3.

Tabela 6.9: Parâmetros estimados para o modelo IPL-Lognormal

Parâmetro	Estimativa (MLE)	IC (90%)	
		LI	LS
K	4,9635E-17	2,8149E-17	8,7522E-17
N	4,7312	4,6418	4,8206
σ	0,0927	0,0764	0,1125

A figura 6.7(a) apresenta o ajuste do modelo IPL-Lognormal aos dados de cada nível de estresse, juntamente com a sua estimativa para o nível de uso (NU). Os valores do eixo x representam o número de requisições HTTP até a falha, em cada nível de estresse. O eixo y representa a probabilidade de falha (*unreliability*). Os pontos (círculos) são os valores observados durante os testes de envelhecimento acelerado para cada nível de estresse.

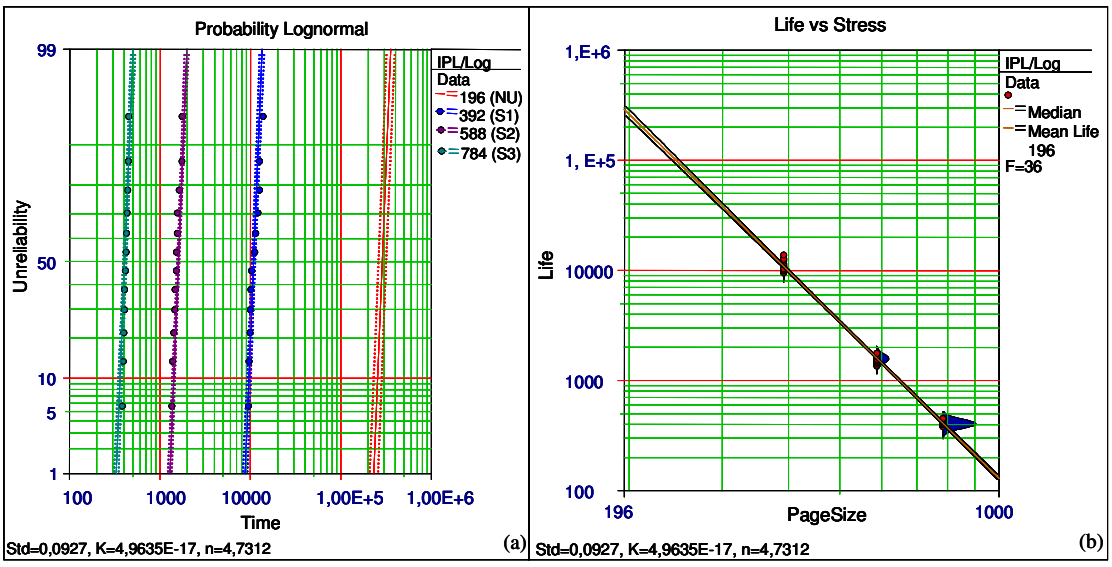

Figura 6.7: (a) Gráfico múltiplo de probabilidade Lognormal com ajuste do modelo IPL-Lognormal; (b) Comportamento da vida média (TTF) estimada em função do estresse

Na figura 6.7(b), tem-se a PDF da IPL-Lognormal ajustada para cada um dos três níveis de estresse, juntamente com sua estimativa para o nível de uso. O eixo x representa o tamanho de página, o qual iniciou com o valor do nível de uso (TMP=196). O eixo y representa o tempo até a falha, ou seja, o tempo de vida do sistema representado em lotes de 100 requisições HTTP. O valor do eixo y, no ponto em que este é cruzado pela reta do modelo IPL (*linearizado*), é o valor médio estimado do tempo de falha para o nível de uso.

A figura 6.8(a) apresenta o gráfico de probabilidade Lognormal do modelo IPL-Lognormal, estimado para o nível de uso com intervalo de 90% de confiança, demonstrando o bom ajuste gráfico deste modelo. Esta avaliação corrobora com a verificação da adequação do modelo pela análise dos resíduos ilustrada na figura 6.8(b).

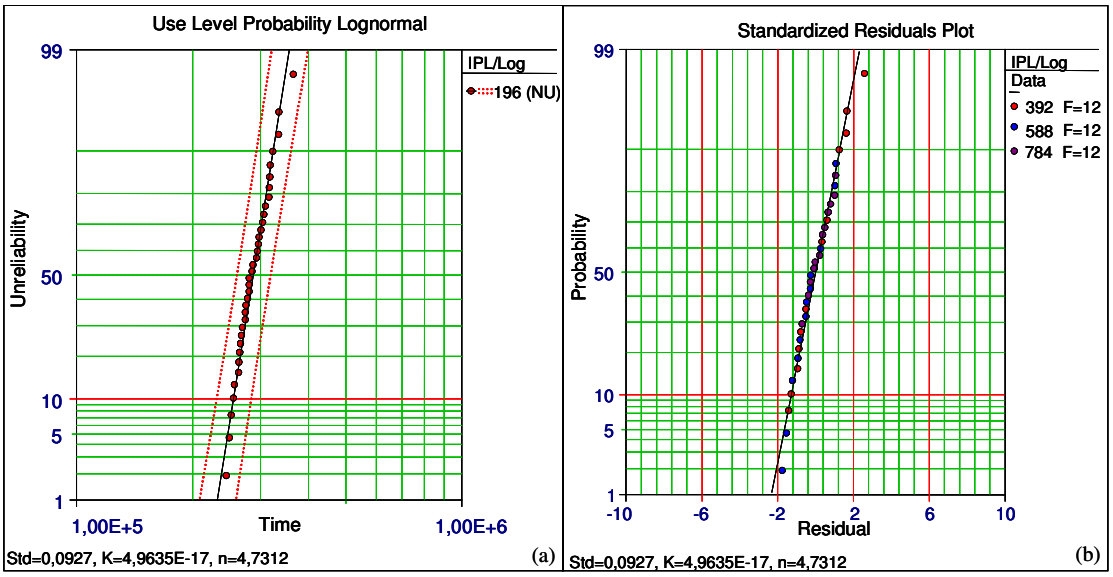

Figura 6.8: (a) Gráfico de probabilidade Lognormal para o nível de uso; (b) Gráfico de probabilidade normal de resíduos do modelo IPL-Lognormal

Como resultado da estimação da distribuição de vida, para o nível de uso, o MTBF calculado para TMP=196 foi de 289.000 lotes de requisições HTTP, cujo IC (90%) foi LI=262.000 e LS=318.770. Considerando que cada lote representou 100 requisições, a estimativa para o número médio de requisições até a falha do servidor web foi de 28.900.000 requisições HTTP.

A figura 6.9(a) apresenta o gráfico da função confiabilidade contra o tempo, obtido a partir do modelo IPL-Lognormal estimado para o nível de uso (TMP=196). A figura 6.9(b) representa a sensibilidade da função confiabilidade em relação ao número de requisições processadas (tempo) e ao estresse de aceleração (fator de envelhecimento).

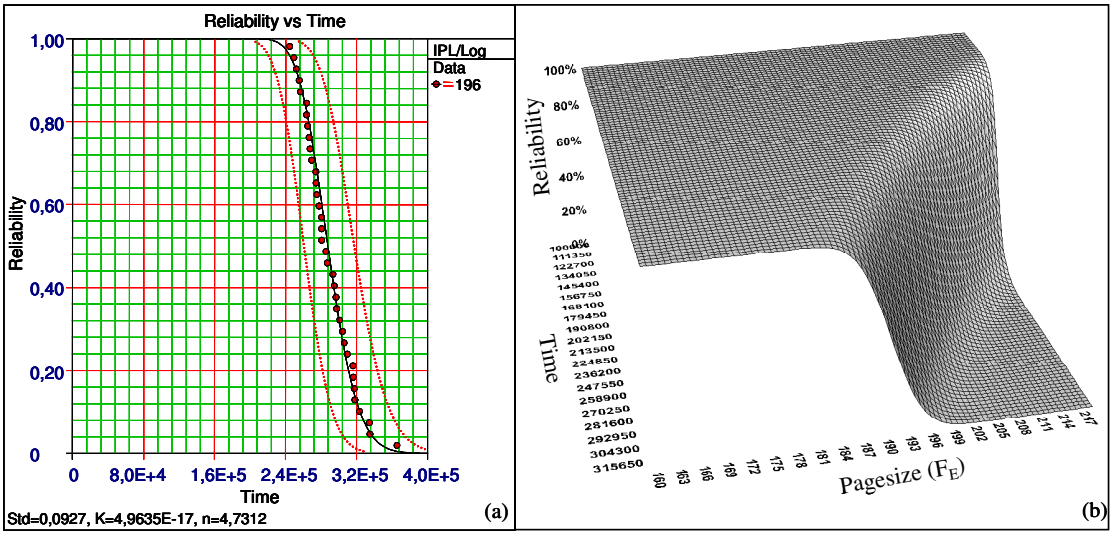

Figura 6.9: (a) Função confiabilidade para o nível de uso (TMP=196); (b) Função confiabilidade contra o tempo e o F_E (tamanho de página dinâmica)

A partir do modelo IPL-Lognormal estimado para o nível de uso, o qual representa a distribuição de vida do servidor web, podem ser calculadas diversas métricas de confiabilidade. Para exemplificar, além do MTBF apresentado anteriormente, foi calculado o total de requisições para páginas dinâmicas que o servidor web pode processar com uma garantia de funcionamento de 99,999%. O resultado foi uma quantidade média de 19.378.000 requisições, cuja estimativa de IC (90%) foi LI = 17.110.000 e LS = 21.947.000.

6.5. ACURACIDADE DO MODELO DE ACELERAÇÃO

No capítulo 1, foi definido como terceiro objetivo específico deste trabalho, analisar quantitativamente a acuracidade das estimativas obtidas com o método proposto, como por exemplo, do MTBF calculado na seção anterior. Para avaliar a acuracidade desta estimativa, seria preciso comparar o MTBF observado durante a operação normal do servidor web (sem acelerar o envelhecimento de software) com o seu valor estimado acelerando o envelhecimento. O problema neste caso é o longo período necessário para observar as falhas por envelhecimento quando o sistema executa em regime normal de operação (sem aceleração).

Para exemplificar o exposto anteriormente, tomou-se o valor do MTBF estimado na seção anterior. Este valor representa o número médio de requisições que o servidor web deve processar para que uma falha por envelhecimento possa ocorrer. Neste caso, para calcular o tempo entre falhas do sistema, deve-se adotar uma taxa de atendimento destas requisições, a

qual seja representativa do nível de uso do servidor web. A fim de se ter um valor de referência, a carga de trabalho do ISP, citado na seção 6.4.1.1, foi caracterizada a fim de obter o tempo médio entre chegadas (TEC) de requisições destinadas a páginas de conteúdo dinâmico. Como já abordado anteriormente, este tipo de requisição segue o padrão definido para o F_E, ou seja, são requisições que ao serem processadas pelo *httpd* ativam uma ou mais faltas que causam o vazamento de memória nos processos do Apache. Os procedimentos para a caracterização de cargas de trabalho não fazem parte do método proposto, portanto estes foram suprimidos desta seção, estando descritos no apêndice C.

O resultado da caracterização da carga de trabalho do ISP, para requisições destinadas a páginas dinâmicas, resultou em um TEC de aproximadamente 9 segundos. Com base neste valor, o MTBF foi convertido (28.900.000 requisições * 9 segundos), resultando em um tempo entre falhas de $\cong 8$ anos. Este longo período de observação é justificado pelo elevado valor do TEC. A explicação para isso deve-se ao pequeno porte do ISP usado como referência para a caracterização do TEC, o qual apresentou um baixo volume de tráfego no período da amostragem. Além disso, vale lembrar que foram consideradas apenas as requisições destinadas para páginas dinâmicas. No caso de um ISP com maior volume de tráfego, cujo TEC fosse, por exemplo, de 0,10 segundo, o tempo de observação seria de $\cong$ 33 dias para a mesma quantidade de requisições. Portanto, o tempo necessário para a observação das falhas depende do TEC das requisições com o padrão do F_E.

Devido à disponibilidade de tempo para a realização desta etapa da pesquisa, o período de observação calculado com o TEC de 9 segundos foi considerado inviável em termos práticos. Como alternativa para viabilizar a observação das falhas, com o objetivo de se calcular a acuracidade das estimativas obtidas com o método proposto, adotou-se a redução do limiar de falha (D_f) de 400 para 100 megabytes. Deste modo, foi possível observar os tempos de falha em todos os níveis de estresse, inclusive no nível de uso, o que permitiu comparar o MTBF estimado com o observado. Todos os procedimentos realizados para D_f = 400 foram realizados para uma nova amostra de tempos de falha obtida com D_f = 100. Neste caso, não existiram tempos de *pseudofalha*, já que o tempo de experimentação foi suficiente para que todos os caminhos de degradação cruzassem o limiar D_f. Os resultados dos experimentos para este cenário estão descritos no apêndice D. Como verificado na tabela 6.10, a estimativa do MTBF demonstrou boa acuracidade em relação ao seu valor observado, o qual se encontrou dentro do intervalo de confiança obtido com a aplicação do método.

Tabela 6.10: MTBF estimado e observado

	MTBF	IC (90%)	
		LI	LS
Estimado (MLE)	365,48	337,92	395,28
Observado	343,57		

Ao considerar apenas os valores médios (estimado e observado), o erro foi de aproximadamente 21,91. Da mesma forma que na seção anterior, os valores do MTBF estão representados em lotes de 100 requisições, o que resulta em uma diferença real de 2.191 requisições HTTP. O MTBF é uma medida de tempo e, portanto, o erro em questão se refere à diferença do tempo médio entre falhas estimado em relação ao observado. Como descrito anteriormente, este tempo depende da carga de trabalho do servidor, o que fica evidente pela simulação de cenários apresentada na tabela 6.11.

Tabela 6.11: Magnitude do erro em minutos

TEC do F_E (segundos)	Erro (minutos)
9	329
5	183
1	37
0,5	18
0,1	4

A segunda linha da tabela apresenta o erro do MTBF estimado, em relação ao valor observado, para o TEC de 9 segundos. As demais linhas são simulações de cenários, que ilustram o tamanho do erro para situações com um menor valor para o TEC. Neste caso, fica evidente que a magnitude do erro está vinculada ao valor do TEC no nível de uso do sistema sob teste. Para o TEC de 9 segundos, o MTBF estimado apresentou um erro de 5,4 horas (acima do valor observado). Vale ressaltar que este cálculo se deu com o valor médio da estimativa. No caso de se adotar este valor de MTBF para a implementação de rotinas de manutenção preventiva, seria prudente assumir uma postura mais conservadora, utilizando como referência o valor do limite inferior do intervalo de confiança estimado (ver tabela 6.10). Neste caso, o erro seria de 1,41 hora abaixo do valor do MTBF observado, o que é bastante aceitável para o cenário em questão.

Complementarmente à quantificação do erro das estimativas, foi realizada uma análise da redução do tempo de experimentação proporcionada pela aplicação do método. O quadro 6.10 apresenta a duração das etapas experimentais do método, durante a realização deste estudo de caso, a fim de comparar com o tempo de observação das falhas por envelhecimento

sem a sua implementação. Os valores são relativos aos experimentos realizados com D_f=400 e D_f=100 megabytes. Para o cenário de D_f=100, vale lembrar que não foi realizado um TEA complementar, pois a amostra piloto foi suficiente (ver apêndice D).

Processo	Duração (min) D_f=400	Duração (min) D_f=100
Seleção do fator de envelhecimento:		
Teste de carga (*httperf*)	480	480
DOE (piloto)	369	369
DOE (definitivo)	3255	3255
Teste de envelhecimento acelerado:		
TEA (piloto)	1288	1056
TEA (complementar)	920	0
Total:	6312	5160

Quadro 6.10: Duração das atividades experimentais do método

Para o primeiro cenário (D_f=400), o tempo total exigido para executar as etapas experimentais do método foi de $\cong$ 6.312 minutos (105,2 horas). O MTBF estimado sem a aceleração do envelhecimento foi de 72.250 horas ($\cong$ 8 anos), o que resultou em uma redução de $\cong$ 687 vezes no tempo de experimentação. Esta comparação não considerou as demais atividades relacionadas com o planejamento dos experimentos e a análise dos seus resultados, dando ênfase apenas nos tempos de experimentação.

No segundo caso (D_f=100), o tempo total para executar as etapas experimentais do método foi de $\cong$ 86 horas. O MTBF observado foi 34.357 requisições, que ao considerar um TEC de 9 segundos, resultou em um tempo de observação de $\cong$ 85,89 horas. Neste caso, não houve diferença significativa entre a utilização do método ou a observação das falhas sem a aceleração do envelhecimento.

6.6. CONSIDERAÇÕES FINAIS

Neste capítulo, foi possível avaliar a aplicabilidade do método proposto, a partir de um estudo experimental de aceleração do envelhecimento aplicado a um software real. Dentre os principais resultados obtidos, primeiramente destaca-se a identificação dos tratamentos que mais contribuíram para o envelhecimento do servidor web Apache usado no estudo de caso. A partir dos fatores investigados, verificou-se que requisições HTTP, destinadas a páginas dinâmicas, foi o padrão de carga de trabalho que mais contribuiu para o envelhecimento dos processos *httpd*, em termos do seu aumento de tamanho na memória. O tamanho das páginas (dinâmicas) foi o segundo fator mais significativo. Além do propósito principal destes fatores,

que foi a aceleração do envelhecimento, estas evidências podem ser usadas em uma análise de causa raiz (RCA), com o objetivo de identificar as faltas, presentes no código do *httpd*, que estejam causando o vazamento de memória. Por exemplo, o tratamento de requisições HTTP destinadas a páginas dinâmicas, geradas por programas CGI, é realizado por um módulo do Apache chamado de *mod_cgi*. Portanto, o código fonte deste módulo poderia ser o ponto de partida para uma RCA com intuito de localizar a origem do vazamento de memória no *httpd*. Neste caso, o método estaria contribuindo para o melhoramento da qualidade do produto durante as fases de desenvolvimento e manutenção.

Outro resultado importante deste estudo de caso, diz respeito à verificação de que a variação na taxa de requisições não influenciou o envelhecimento dos processos *httpd*. Esta constatação é relevante, na medida que trabalhos publicados recentemente (BAO; SUN; TRIVEDI, 2005; XIE; HONG; TRIVEDI, 2004) têm considerado que este fator tem influência significativa no envelhecimento de software. A partir das evidências experimentais, demonstradas neste capítulo, que corroboraram com a discussão teórica do capítulo 2, fica claro que apenas a variação na taxa de chegada não pode ser considerada um fator de influência para o envelhecimento. Se as requisições que chegam no sistema não possuem o padrão de ativação das faltas relacionadas ao envelhecimento (FRE), estas por sua vez não exercem qualquer influência sobre o envelhecimento dos processos, independente da sua taxa de chegada. Como exemplo deste comportamento, tem-se o caso do tratamento T4, durante a seleção do fator de envelhecimento (ver seção 6.4.1), que apesar de sua carga de trabalho exigir 90% da capacidade do servidor web, não teve qualquer efeito em termos do envelhecimento do *httpd*. Até o momento, não se tem registro de trabalhos abordando o problema do envelhecimento de software seguindo uma abordagem orientada à suas causas, tendo sido mais freqüente, publicações de estudos realizados à cerca dos efeitos do envelhecimento.

A aceleração da degradação dos recursos de memória principal do servidor web, a partir do teste de envelhecimento acelerado, juntamente com sua posterior análise para a obtenção da distribuição de vida do sistema investigado, também podem ser consideradas resultados importantes deste capítulo. Na literatura revisada, não foram encontrados trabalhos voltados para a aceleração da degradação de sistemas de software, o que configura uma aplicação pioneira nesta área.

Com relação à acuracidade do método, verificou-se que o erro das estimativas depende do tempo entre chegadas das requisições que seguem o padrão do fator de envelhecimento. Também, é possível concluir que a utilização do método é viável apenas nos casos onde a

observação das falhas por envelhecimento excede o tempo necessário para realizar a experimentação, como foi o caso do primeiro cenário ($D_f = 400$). De fato, esta é uma premissa para a utilização das técnicas de ALT e ADT utilizadas pelo método. Dentre os estudos experimentais revisados no capítulo 3, em todos os casos, o tempo de experimentação necessário para detectar os efeitos do envelhecimento foi superior ao tempo total de experimentação do estudo de caso apresentado neste capítulo. Esta diferença pode ser verificada comparando os valores listados no quadro 3.3, cujo menor valor reportado foi de 14 dias, com o tempo total de $\cong 4.4$ dias (105,2 horas) utilizado pelas etapas experimentais deste trabalho (cenário com $D_f = 400$). Ressalta-se que a duração de 14 dias, reportada no trabalho de Huang *et al.* (1995), foi para a detecção dos indícios de envelhecimento e não para a observação das falhas por envelhecimento, o que exigiria um período maior de experimentação. Dentre as pesquisas listadas no quadro 3.3, o trabalho de Li, Vaidyanathan e Trivedi (2002) foi aquele que apresentou a maior similaridade com o estudo de caso deste capítulo. Naquele trabalho, os autores reportaram uma duração de aproximadamente 46 dias em sua etapa experimental, que da mesma forma como no trabalho de Huang *et al*., refere-se ao período de detecção dos efeitos do envelhecimento e não de observação de falhas. Mesmo assim, este período se comparado com o tempo de experimentação do estudo de caso apresentado, indica uma redução, da etapa experimental, superior a dez vezes em favor do método proposto.

CAPÍTULO 7

CONCLUSÕES DA PESQUISA

7.1. RESULTADOS OBTIDOS

O objetivo geral desta pesquisa foi propor uma abordagem sistematizada para acelerar as falhas de sistemas causadas por envelhecimento de software. O método proposto para alcançar este objetivo foi descrito no capítulo 5, sendo subsidiado pelo referencial teórico tratado nos capítulos anteriores. A verificação da aplicabilidade do método foi realizada, com a condução do estudo experimental apresentado no capítulo 6, cujos resultados demonstraram que os objetivos específicos foram atingidos de forma satisfatória.

O primeiro objetivo específico foi alcançado, ao se obter evidências experimentais de que a variação do fator de envelhecimento permitiu acelerar o envelhecimento dos processos *httpd*. O segundo objetivo específico também foi considerado atendido, na medida que o modelo de potência inversa (IPL) foi avaliado em termos da sua aderência à amostra de tempos de falha e *pseudofalha*, os quais foram obtidos com a aceleração do envelhecimento dos processos do *httpd*. Os resultados desta avaliação demonstraram que o IPL foi adequado para modelar os dados de vida obtidos com a aceleração do envelhecimento do *httpd*. Contudo, ressalta-se que a utilização deste modelo pode não ser adequada para experimentos envolvendo outros componentes/produtos de software. Para estes casos, seria necessária a adoção de novos modelos utilizados na área de ensaios acelerados, ou até mesmo definidos empiricamente. O bom ajuste apresentado pelo modelo IPL foi positivo, pois serve de ponto de partida para o teste de novos modelos, já que, na literatura, não se tem registro da utilização de modelos de relacionamento vida-estresse aplicados a dados de degradação acelerada em produtos e/ou componentes de software. Com respeito ao terceiro objetivo específico, a avaliação da acuracidade realizada com os resultados do estudo de caso, produziu evidências positivas à cerca da qualidade das estimativas obtidas com o método proposto. Vale ressaltar, que novos experimentos são necessários a fim de diversificar o modelo de aceleração e os produtos de software avaliados, objetivando estender a abrangência da análise dos processos definidos pelo método. Os resultados positivos, apresentados no

estudo experimental do capítulo 6, são evidências que motivam a realização de novos experimentos neste sentido.

7.2. CONTRIBUIÇÃO PARA A LITERATURA NA ÁREA

Durante o desenvolvimento desta pesquisa, seus resultados foram progressivamente comunicados na forma de artigos e palestras. Os artigos publicados estão listados no quadro 7.1. Apesar do ISSRE (www.issre.org) ainda não ter sido classificado pelo Qualis/CC, este é considerado, atualmente, o principal evento científico internacional na área de engenharia de confiabilidade de software.

Ano	Evento	Classificação QUALIS/CC	Referência
2004	WPERFORMANCE	Nacional B	Matias Jr *et al*. (2004)
2004	ESELAW	NC	Matias Jr e Freitas Filho (2004)
2005	ISSRE/IP	NC	Matias Jr *et al*. (2005a)
2005	ESELAW	NC	Matias Jr *et al*. (2005b)
2006	COMPSAC	Internacional A	Matias Jr e Freitas Filho (2006)
NC – Não está classificado no sistema QUALIS Ciência da Computação.			

Quadro 7.1: Artigos publicados

Em Matias Jr. *et al*. (2004) foi implementado e verificado experimentalmente o primeiro processo do método proposto. Ressalta-se que neste artigo, a etapa do DOE não foi implementada com replicações. O trabalho Matias Jr e Freitas Filho (2004), dedicou-se a apresentar a abordagem, baseada em DOE, para caracterizar o envelhecimento de software a partir da identificação dos fatores de ativação das faltas relacionadas ao envelhecimento. Em Matias Jr e Freitas Filho (2006) os experimentos do primeiro artigo foram reproduzidos, contudo implementando replicações na etapa do DOE além da adoção de novos mecanismos mais acurados para a monitoração do envelhecimento dos processos *httpd*. Em Matias Jr. *et al*. (2005a) e Matias Jr. *et al*. (2005b), são apresentados os resultados obtidos com a implementação de um modelo para simular o envelhecimento de software em um servidor web. O modelo de simulação foi implementado com base em resultados de testes de sobrecarga de um sistema servidor web similar ao ambiente utilizado pelo primeiro trabalho. A partir dos resultados da simulação, vários modelos de distribuição de vida foram testados contra a amostra de tempos de falha, e verificou-se que o melhor ajuste foi de uma distribuição Weibull (triparamétrica). O MTTF estimado foi de aproximadamente 75 dias e a

simulação de cinqüenta falhas foi realizada em 11 horas[35]. Em primeira análise, a utilização da simulação para obter os tempos de falha em menor tempo, seria uma alternativa ao método de aceleração de envelhecimento proposto neste trabalho. Contudo, a construção do modelo de simulação depende de uma etapa experimental para caracterizar o comportamento do envelhecimento no sistema que se pretende simular. Se nesta etapa a observação dos efeitos do envelhecimento exigir um longo período de experimentação, além das replicações dos testes para se estabelecer uma determinada confiança das estimativas, a construção do modelo de simulação pode ser inviável dependendo das restrições de tempo e recursos do experimentador. Neste ponto, destaca-se a importância do método proposto, com o objetivo de reduzir o tempo da etapa experimental. Todos os trabalhos citados foram desenvolvidos com o intuito de validar o enfoque dado ao primeiro processo método, haja vista ser este a base para os demais processos.

Complementarmente aos artigos publicados, alguns dos resultados desta pesquisa foram também divulgados na forma de palestras, as quais estão listadas no quadro 7.2.

Ano	Título	Evento	Cidade
2005	Apache em sites de Alto Tráfego: Aspectos internos de performance e escalabilidade.	6º Fórum Internacional de Software Livre †	Porto Alegre/RS
2006	Apache Reliability: Mitigando os efeitos de *memory leaks* no *httpd.*	7º Fórum Internacional de Software Livre ‡	Porto Alegre/RS
2006	Utilizando Ensaios Acelerados para a Análise de Confiabilidade de Servidores Web com Sintomas de Envelhecimento de Software.	The International Applied Reliability Symposium - South America Edition †	Salvador/BA

† http://chopin.softwarelivre.org/papers/pub/
‡ http://fisl.softwarelivre.org/7.0/papers/pub/
† http://www.arsymposium.org/southamerica/2006/2006matrix_t.htm

Quadro 7.2: Palestras realizadas

A partir das comunicações supracitadas, entende-se que as contribuições deste trabalho se mostram relevantes, na medida que seus resultados têm sido avaliados em âmbito nacional e internacional. Também, a partir da divulgação dos seus resultados, espera-se que outros pesquisadores possam replicar os experimentos realizados neste estudo, assim como desenvolver novos experimentos, de forma a expor o método a outros cenários de carga de trabalho e classes de aplicações de software.

[35] Utilizando um processador (Celeron, 1.2 Ghz, 512 megabytes de RAM) em execução dedicada.

7.3. DIFICULDADES ENCONTRADAS

Uma das principais dificuldades encontradas diz respeito à ausência de estudos na área de ensaios de vida acelerados e ensaios de degradação acelerados aplicados a produtos/componentes de software. A falta de pesquisas propondo e/ou avaliando experimentalmente modelos de relacionamento vida-estresse voltados para produtos de software, exigiu um maior esforço durante a etapa de definição do modelo adotado.

Outras duas dificuldades que merecem destaque referem-se à etapa experimental. A primeira refere-se à necessidade de automatização das rotinas de testes, haja vista as suas execuções por várias horas de forma ininterrupta. Neste contexto, o maior esforço esteve associado com a sincronização do gerador de carga (*httperf*), do lado cliente, com o monitor dos processos (*sysmon*) do lado servidor. A outra dificuldade, em nível experimental, foi com relação à precisão da monitoração do tamanho dos processos no Linux. Existem várias ferramentas para este propósito (ex. *top*, *ps*), contudo, os programas avaliados não se mostraram adequados, principalmente pela falta de precisão e consistência nos dados referentes ao tamanho dos processos, ou também por não se adequarem à infra-estrutura automatizada desenvolvida para o trabalho. Devido a isso foi criado o software *sysmon* que serviu a este propósito.

Por se tratar de um trabalho de experimentação, envolvendo replicações, ressalta-se o esforço e atenção necessários para se garantir o mesmo ambiente de teste para cada replicação, tanto durante o DOE (1º processo) quanto no TEA (2º processo), o que muitas vezes exigiu sacrificar testes realizados por várias horas a fim de garantir este requisito. Estes casos ocorreram basicamente em decorrência da execução automatizada dos testes, o que foi resolvido com o aperfeiçoamento da instrumentação utilizada.

Vale também destacar a importância da infra-estrutura de energia que alimenta os recursos computacionais dedicados aos experimentos. Recomenda-se fortemente a adoção de uma fonte alternativa de energia, haja vista que os dados de experimentos de longa duração podem ser comprometidos em casos de quedas de energia ou variações de tensão na rede, o que normalmente causa a reinicialização não programada dos equipamentos. Fatos como estes, ocorreram nos estágios iniciais desta pesquisa, que posteriormente foram reduzidos com a utilização de *nobreaks*. Este é um fator importante, haja vista que o principal objetivo do método é a redução do tempo e custos de experimentação durante a observação das falhas causadas por envelhecimento de software. Uma elevada freqüência de problemas desta natureza, pode comprometer os benefícios do método no que se refere à redução do tempo de

experimentação, tendo em vista a necessidade de repetição de experimentos de longa duração encerrados abruptamente por tais problemas.

7.4. FUTUROS TRABALHOS

Os resultados obtidos no capítulo 6, apesar de serem evidências positivas sobre a aplicabilidade do método proposto, não são suficientes para se fazer generalizações ao seu respeito. A replicação dos experimentos descritos no capítulo 6, por outros pesquisadores, a fim de se revisar os processos do método, bem como reproduzir os resultados obtidos, é o caminho natural para o amadurecimento da abordagem proposta.

Posteriormente, sugere-se novas aplicações do método envolvendo outros tipos de software. São candidatos naturais, sistemas de gerenciamento de bases de dados (SGBD), sistemas de arquivos, sistemas de simulação computacional, sistemas de controle de processos industriais, software embarcado em dispositivos de missão crítica (ex. centrais telefônicas), dentre outras aplicações. Todas estas áreas têm em comum, a grande longevidade de seus processos de software executando de forma ininterrupta, sendo cenários propícios para a manifestação dos efeitos do envelhecimento de software. Não é incomum que grandes sistemas de armazenamento, tais como aqueles utilizados em ambientes de telecomunicações ou centro de dados (*data centers*), sofram de perda gradativa de desempenho devido à fragmentação de seus sistemas de arquivos, que é considerado um efeito causado pelo envelhecimento de software. Nestes casos, a aplicação do método teria como objetivo determinar os tempos em que manutenções preventivas (ex. rotinas de *desfragmentação*) seriam realizadas, a fim de evitar que o desempenho do sistema atinja níveis de serviço considerados inaceitáveis. Também, vale destacar que em ambientes de computação distribuída de alto desempenho (DANTAS, 2005), devido à existência de processos executando ininterruptamente por longos períodos de tempo, tem-se uma maior propensão para ocorrências do envelhecimento de software. Nestes casos, o método se aplicaria da mesma forma como no estudo de caso do servidor web, haja vista que o envelhecimento seria tratado para cada processo individualmente. Outro cenário a se considerar, seria a utilização da computação de alto desempenho para implementar a aceleração do envelhecimento de um dado sistema por meio da elevação da taxa de uso (ver seção 4.2.3.1).

Como já citado anteriormente, uma evolução desta pesquisa está na realização de testes com novos modelos de relacionamento vida-estresse em alternativa ao modelo IPL. A diversidade de cenários, aos quais o método pode ser aplicado, sem dúvida exigirá outros

modelos que capturem os efeitos do envelhecimento, em função dos fatores presentes nas diversas cargas de trabalho existentes nestes novos ambientes de aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT - ASSOCIACAO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 9000-4**: **Normas para a gestão de qualidade e garantia da qualidade. Parte 4 - Guia para gestão do programa de dependabilidade**. Rio de Janeiro: ABNT, 1993. 5p. ABNT/CB-25. Equivalência: ISO 9000-4/93.

ABNT - ASSOCIACAO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 5462**: **Confiabilidade e mantenabilidade - terminologia**. Rio de Janeiro: ABNT, 1994. 37p. ABNT/CB-03:056.01. Equivalência: IEC 50 191.

ADAMIC, L. A.; Huberman, B. A. Zipf's Law and the Internet. **Glottometrics**, vol. 3, p. 143-150, 2002.

AVIŽIENIS, A.; LAPRIE, J.; RANDELL, B. **Fundamental concepts of dependability**, Relatório Técnico N01145, LAAS-CNRS, 2001.

AVIŽIENIS, A.; LAPRIE, J.; RANDELL, B.; LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. **IEEE Transactions on Dependable and Secure Computing**, v. 1, n.01, p. 11-33, jan-mar. 2004.

AVRITZER, A.; WEYUKER, E. J. A Monitoring smoothly degradating systems for increased dependability. **Empirical Software Engineering Journal**, v. 2, n. 1, p. 59-77, 1997.

BAO, Y.; SUN, X.; TRIVEDI, K.S. A workload-based analysis of software aging, and rejuvenation. **IEEE Transactions on Reliability**, 3. ed., v. 54 , p. 541 – 548, set. 2005.

BARBETTA, P. A.; REIS, M. M.; BORNIA, A. C. **Estatística para Cursos de Engenharia e Informática**, Editora Atlas, 2004.

BASILI, V. R.; PERRICONE, B. T. Software errors and complexity: an empirical investigation. **Communications of the ACM**, v. 27, n. 1, p. 42-52, 1984.

BOBBIO, A.; SERENO, A.; ANGLANO, C. Fine grained software degradation models for optimal software rejuvenation policies. **Journal of Performance Evaluation**, v. 46, n. 1, p. 45-62, 2001.

BROOKS, F. P. Jr. No Silver Bullet: Essence and accidents of software engineering. **IEEE Computer**, USA, v. 20, n.4 , p. 10-19, mar. 1987.

BROWN, A.; PATTERSON, D. A. Embracing failure: a case for recovery-oriented computing (ROC). In: **Proceedings of the 2001 High Performance Transaction Processing Symposium** (HPTS '01), USA, 2001.

CANDEA, G. **The Enemies of Dependability I**: Software. Notas de aula – CS444a. Disponível em: <http://www.stanford.edu/~candea/teaching/cs444a-fall-2003/notes /software.pdf>. Acesso em: 01 out. 2005.

CASTELLI, V.; HARPER, R. E.; HEIDELBERGER, P.; HUNTER, S. W.; TRIVEDI, K. S.; VAIDYANATHAN , K.; ZEGGERT, W. P. Proactive management of software aging. **IBM Journal of Research & Development**, v. 45, n. 2, mar. 2001.

CHANDRA, S.; CHEN, P.M. How fail-stop are faulty programs? In: **Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing**, 1998. p. 240-249.

CHILLAREGE, R.; KAO, W.; CONDIT, R. G. Defect type and its impact on the growth curve. In: **International Conference on Software Engineering**, 1991, p. 246-255.

CHILLAREGE, R., GOSWANI, K., DEVARAKONDA, M. Experiment illustrating failure acceleration and error propagation in fault-injection. In: **International Symposium on Software Reliability Engineering**, 2002.

CHOU, A.; YANG, J.; CHELF, B.; HALLEM, S.; ENGLER, D. An empirical study of operating systems errors. In: **Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles**, 2001.

CISCO SYSTEMS. **Cisco Security Advisory**: Cisco Catalyst memory leak vulnerability. ID: 13618, 2001. Disponível em: <http://www.cisco.com/warp/public/707/catalyst-memleak-pub.shtml>. Acesso em: 01 nov. 2004.

COLEMAN, D. E.; MONTGOMERY, C. D. A systematic approach to planning for a designed industrial experiment, **Technometrics**, vol. 35, n. 01, 1993.

DANTAS, M. Computação Distribuída de Alto Desempenho. Axcel Books, 2005.

DAOUDI, K.; VÉHEL, J. L.; MEYER, Y. Construction of continuous functions with prescribed local regularity. **Journal of Constructive Approximation**, v.14, n. 3, p. 349-385, 1998.

DERSHOWITZ, N. **Software Horror Stories**. Disponível em: <http://www.cs.tau.ac.il/~nachumd/horror.html>. Acesso em: 28 jun. 2005.

DOHI, T.; GOSEVA-POPSTOJANOVA, K.; TRIVEDI, K.S. Analysis of software cost models with rejuvenation. In: **IEEE International Symposium on High Assurance Systems Engineering**, 5., 2000, HASE. Alburquerque, USA, 2000. p. 25-34.

DRAPER, N. R.; SMITH, H. **Applied Regression Analysis**., ed. 02, New York: John Wiley & Sons, 1981.

EHRLICH, W.; NAIR, V.N.; ALAM, M.S.; CHEN, W.H.; ENGEL, M. Software reliability assessment using accelerated testing methods, **Journal of the Royal Statistical Society**, vol 47, n. 1, 1998, p. 15-30.

FENTON, N. E.; OHLSSON, N. Quantitative analysis of faults and failures in a complex software system. **IEEE Transactions on Software Engineering**, v. 26, n. 8, p. 797-814, 2000.

FERREIRA, A. B. H. **Novo Dicionário Aurélio da Língua Portuguesa.** ed. 03, Editora Positivo, 2004.

FREITAS, M. A.; COLOSIMO, E. A. Confiabilidade: análise de tempo de falha e testes de vida acelerados. In: **Série Ferramentas da Qualidade**, v.12, Fundação Christiano Ottoni, Belo Horizonte, UFMG, 1997.

FREITAS FILHO, P. J. **Introdução à modelagem e simulação de sistemas com aplicações em Arena**, Visual Books, 2001.

FULTON, N. D.; HUANG, Y.; KINTALA, C. M. R.; KOLETTIS, N. J. **Apparatus and methods for software rejuvenation.** USA n. PI 5.715.386, 1996.

GANAPATHI, A.; WANG, Y.; LAO, N.; WEN, J. Why PCs are fragile and what we can do about It: A study of windows registry problems. In: **Proceedings of the 2004 International Conference on Dependable Systems and Networks** (DSN'04), Itália, 2004, p. 561.

GANAPATHI, A.; PATTERSON, D. Crash data collection: a windows case study. In: **Proceedings of International Conference on Dependable Systems and Networks**, Japão, 2005, p. 280-285.

GENERAL ACCOUNTING OFFICE (GAO) - Information management and technology division. Relatório Técnico **GAO/IMTEC-92-26: Patriot Missile Software Problem.** USA, 1992.

GARG, S.; HUANG, Y.; KINTALA, C.; TRIVEDI, K. S. Time and load based software rejuvenation: policy, evaluation and optimality. In: **Proceedings of the First Fault-Tolerant Symposium**, India, 1995a.

GARG, S.; PULIAFITO, A.; TELEK, M.; TRIVEDI, K. Analysis of software rejuvenation using markov regenerative stochastic petri net. In: **Proceedings of the Sixth International Symposium of Software Reliability**, 6., p. 180-187, 1995b.

GARG, S.; HUANG, Y.; KINTALA, C.; TRIVEDI, K. S. Minimizing completion time of a program by checkpointing and rejuvenation. In: **Proceedings of the ACM SIGMETRICS Conference**, 1996, Philadelphia, p. 252–261, 1996a.

GARG, S.; PFENING, A.; PULIAFITO, A.; TELEK, M.; TRIVEDI, K.S. Modeling and analysis of load and time dependent software rejuvenation policies. In**: Proceedings of the International Workshop on Performability Modeling of Computer and Communication Systems**, p. 35-39, 1996b.

GARG, S.; MOORSEL, A.; VAIDYANATHAN, K.; TRIVEDI, K. A methodology for detection and estimation of software aging. In: **Proceedings of the 9th International Symposium on Software Reliability Engineering**, Alemanha, p. 282-292, 1998a.

GARG, S.; PULIAFITO, A.; TELEK, M.; TRIVEDI, K. Analysis of preventive maintenance in transactions based software systems. **IEEE Transactions on Computers**, v. 47, n. 1, p. 96-107, 1998b.

GRAY, J. Why do computers stop and what can be done about it? In: **Proceedings of the Symposium on Reliability in Distributed Software and Database Systems**, 1986, p. 3-12.

GROSS, K. C.; WEGERICH, S.W.; SINGER, R. M. New artificial intelligence technique detects instrument faulty early. **Power Magazine**, McGraw-Hill, v. 42 n. 06, p. 89-95, 1998.

GROSS, K. C.; BHARDWAJ, V.; BICKFORD, R. Proactive detection of software aging mechanisms in performance critical computers. In: **Proceedings of the 27th Annual NASA GODDARD/IEEE Software Engineering Workshop**, NASA, USA, 2002, p.17 – 23.

HANKE, J. E.; REITSCH, A. G.; WICHERN, D. W. **Business Forecasting**, ed. 07, Prentice Hall, USA, 2001.

HARPER, R. E.; HUNTER, S. W. **Method and system for transparent time-based selective software rejuvenation**. USA n. PI 6.594.784, 1999.

HARPER, R. E.; HUNTER, S. W; PAHEL, JR.; THOMAS D. **Method and system for transparent symptom-based selective software rejuvenation.** USA n. PI 6.629.266, 1999.

HARPER, R. E.; HUNTER, S. W. **System and method for performing automatic rejuvenation at the optimal time based on work load history in a distributed data processing environment**. USA n. PI 6.820.215, 2004.

HECHT, H. Rare conditions: an important cause of failures. In: **Proceedings of the Eighth Annual Conference on Computer Assurance.** IEEE Computer Society, USA, p. 81-85, 1993.

HUANG, Y.; KINTALA, C. M. R.; KOLETTIS, N.; FULTON, N. D. Software rejuvenation: analysis, module and applications. In: **25th International Symposium on Fault Tolerant Computing** (FTCS-25), IEEE Computer Society Press, USA, p. 381-90, 1995.

HUANG, Y.; KINTALA, C.M.R.; BERNSTEIN, L.; WANG, Y. Components for software fault tolerance and rejuvenation. **AT&T Technical Journal**, USA, v. 75, n. 2, 1996, p. 29-37.

IBM. **JR13709: SYSTEM HANG ON DOSISEMWAIT DUE TO UNRELEASED SEMAPHORE IN QUECALLS**. IBM Software Group,2000. Disponível em: <http://www-1.ibm.com/support/docview. wss?uid=swg1JR13709> Acesso em 13 nov. 2005.

IBM. **IY62977: HEAP CORRUPTION RPC MT LIBRARY.** IBM Server Group, 2004. Disponível em: <http://www-1.ibm.com/support/docview.wss?uid=isg1IY62977> Acesso em: 22 nov. 2005.

IBM. **FIN_WAIT_2 sessions on HP-UX from tsm client session**. IBM Software Group, 2005. Disponível em: <http://www-1.ibm.com/ support/docview.wss?rs=0&q1= FIN_WAIT&uid=swg21140123&loc=pt_PT&cs=utf-8&cc=pt&lang=pt+en> Acesso em: 15 nov. 2005.

IEEE. **IEEE Standards Collections: Software Engineering**, IEEE Standard 610.12-1990, IEEE, USA, 1990.

JAIN, R. **The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling**. John Wiley & Sons, USA, 1991.

JALOTE, P. **Fault Tolerance in Distributed Systems**. New Jersey: Prentice-Hall, 1994.

JALOTE, P.; MURPHY, B.; GARZIA, M. R.; ERREZ, B. Measuring reliability of software products. In: **IEEE International Symposium on Software Reliability Engineering**, França. 2004.

LEVESON, N.; TURNER, C. S. An investigation of the therac-25 accidents**. IEEE Computer**, USA, v. 26, n. 7, p. 18-41, jul. 1993.

LI, L.; VAIDYANATHAN, K.; TRIVEDI, K. S. An approach for estimation of software aging in a web server. In: **International Symposium on Empirical Software Engineering**, Japão, 2002, p. 91-100.

LI, W. **The Zipf's Law**, Disponível em: http://www.nslij-genetics.org/wli/zipf/. Acesso em: 01 nov. 2005.

LIU, Y.; MA, Y.; HAN, J. J.; LEVENDEL, H.; TRIVEDI, K.S. Modeling and analysis of software rejuvenation in cable modem termination systems, In: 13th **International Symposium on Software Reliability Engineering** (ISSRE'02), USA, 2002.

MALLOWS, C. L. Some comments on CP. **Technometrics,** v. 42, n. 1, 2000. p. 87-94.

MARSHALL, E. Fatal error: how patriot overlooked a scud. ***Science***, 1992. p. 1347.

MATIAS JR., R.; FREITAS FILHO, P.J., Software aging characterization based on a DOE approach, **1st Experimental Software Engineering Latin American Workshop**, Brasília/DF, 2004.

MATIAS JR., R.; MARCHIORO, E.; SPECIALSKI, E. S.; FREITAS FILHO, P. J., Implementação e avaliação experimental de um agente de rejuvenescimento de software para servidores web, **III WPERFORMANCE - Workshop em Desempenho de Sistemas Computacionais e de Comunicação**, Salvador/BA, 2004.

MATIAS JR., R.; FREITAS FILHO, P. J.; GUEDES, L.; DIAS, A. Reliability estimation of web servers under software aging effects. In: **16th IEEE International Symposium on Software Reliability Engineering, Industry Practices Program**, USA, 2005a.

MATIAS JR., R.; FREITAS FILHO, P. J.; GUEDES, L.; DIAS, A. Estimation of web servers' reliability with symptoms of software aging. In: **2nd Experimental Software Engineering Latin American Workshop**, Uberlândia/MG, 2005b.

MATIAS JR., R.; FREITAS FILHO, P. J. In: **30th IEEE Annual International Computer Software and Applications Conference**, USA, 2006, No prelo.

MEEKER, W. Q., ESCOBAR, L. A. **Statistical Methods for Reliability Data**, USA, 1998. ISBN 0-471-14328-6.

MEEKER, W. Q.; ESCOBAR, L. A.; LU, C. J. Accelerated degradation tests: modeling and analysis. **Technometrics**, v. 40, n. 2, 1998.

MENASCÉ, D. Web server software architectures. **IEEE Internet Computing**, USA, p. 78-81, dec. 2003.

METTAS, A. Understanding accelerated life testing analysis, **The International Applied Reliability Symposium - South America Edition**, Rio de Janeiro, 2003, p. 1-16.

MONTGOMERY, D. C.; RUNGER, G. C. **Estatística Aplicada e Probabilidade para Engenheiros**. ed. 02., LTC, 2003.

MONTGOMERY, D. C. **Design and Analysis of Experiments**. 6. ed. USA: John Wiley, 2005.

MOSBERGER, D.; JIN, T. httperf – A tool for measuring web server performance, In: **First Workshop on Internet Server Performance**, USA, 1998.

MUSA, J. D.; IANNINO, A.; OKUMOTO, K. **Software Reliability: Measurement, Prediction, Application**. McGraw-Hill, USA, 1987.

NELSON, W. Analysis of performance-degradation data from accelerated tests, **IEEE Transactions on Reliability**, vol. R-30, n. 02, 1981.

NELSON, W. **Accelerated Testing: Statistical Models, Test Plans, and Data Analysis.** Ed. 02, USA, 2004. ISBN: 0-471-69736-2.

NETCRAFT. **Web server Survey**, Disponível em: http://news.netcraft.com/archives/ web_server_survey.html. Acesso em: 17 junho 2006a.

NETCRAFT. **Site Report**, Disponível em: http://toolbar.netcraft.com/site_report. Acesso em: 20 junho 2006b.

NETER, J., KUTNER, M. H., NACHTSHEIM, C. J., WASSERMAN, W. **Applied Linear Regression Models**, ed. 03, Richard Irwin, Inc., USA, 1996.

NEUMANN, P. G. **Computer Related Risks**. Addison-Wesley, reading ACM Press, 1995.

PARNAS, D. L. Software aging. In: **Proceedings of the 16th International Conference on Software Engineering** (ICSE´07), 1994, Itália, p. 279-287.

PROJECT MANAGEMENT INSTITUTE (PMI). **Guide to the project management body of knowledge (PMBOK)**, ed. 03, USA, 2004.

PRESSMAN, R. S. **Software Engineering: A practitioner's Approach**, ed. 06, The McGraw-Hill Companies, Inc., USA, 2005.

QIN, F.; TUCEK, J.; ZHOU, Y. Treating bugs as allergies: a safe method for surviving software failures. In: **Proceedings of the USENIX Tenth Workshop on Hot Topics in Operating Systems** (HotOS'05), USA, 2005.

RELIASOFT, A simple demonstration of accelerated life testing analysis, **Reliability Edge**, vol. 2, ed. 01, p. 4-5, 2001.

SHERESHEVSKY, M.; CUKIC, B.; CROWEL , J.; GANDIKOTA , V.; LIU, Y. Software aging and multifractality of memory resources. In: **The International Conference on Dependable Systems and Networks**, 2003, San Francisco, USA. P. 721-730.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. ed. 03,**USA: Addison-Wesley, 1998.

STALLINGS, W. **OPERATING SYSTEMS: Internals and design Principles**. Ed. 05, USA: Prentice Hall, 2005.

SULLIVAN, M.; CHILLAREGE, R. Software defects and their impact on system availability-a study of field failures in operating systems. In: **Proceedings of the 21st IEEE International Symposium on Fault-Tolerant Computing**, 1991, p. 2-9.

TAI, A.T.; ALKALAI, L. On-board maintenance for long-life systems. In: **IEEE Workshop on Application - Specific Software Engineering and Technology**, 1998, p. 69.

TECHNET. **IIS 5.0 Process Recycling Tool.** Microsoft, Jun. 2001. Disponível em: <http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/downloads/iis5rweb.mspx>. Acesso em: 6 ago. 2005.

THE APACHE GROUP. **Apache httpd Project.** Disponível em: <http://httpd.apache.org>. Acesso em: 15 dez. 2005.

TORRES-POMALES, W. **Software Fault Tolerance: A Tutorial. NASA Langley Research Center**, NASA/TM-2000-210616, out. 2000.

TRACHTENBERG, M. Why failure rates observe zipf' s law in operational software. **IEEE Transactions on Reliability**, v. 41, n. 31, p. 386-389, 1992.

TRIVEDI, K. S.; VAIDYANATHAN, K.; GOSEVA-POPSTOJANOVA, K. Modeling and analysis of software aging and rejuvenation. In: 33rd ANNUAL SIMULATION SYMPOSIUM, 2000, p. 270.

TRIVEDI, K. S. **Probability and Statistics with Reliability, Queuing, and Computer Science Applications**. USA, John Wiley and Sons, 2001.

TRIVEDI, K. S. **Duke University**. Disponível em: <http://srejuv.ee.duke.edu/>. Acesso em: jan. 2004.

VAIDYANATHAN, K.; TRIVEDI, K. S. A measurement-based model for estimation of resource exhaustion in operational software systems. In: **Proceedings of the Tenth IEEE International Symposium on Software Reliability Engineering**, USA, 1998, p. 84–93.

VAIDYANATHAN, K.; TRIVEDI, K. S. A measurement-based model for estimation of resource exhaustion in operational software systems. In: **International Symposium on Software Reliability Engineering**, USA, 1999, p. 84.

VAIDYANATHAN, K.; TRIVEDI, K. S. Extended classification of software faults based on aging. Fast abstracts, In: **Proceedings of the 12th International Symposium on Software Reliability Engineering**, 2001a, Hong Kong.

VAIDYANATHAN, K.; TRIVEDI, K. S. Workload-based estimation of resource exhaustion in software systems. In: **International Workshop on Performability Modeling of Computer and Communication Systems**, 5., Alemanha, 2001b. p. 15-16.

VAIDYANATHAN, K.; HARPER, R. E.; HUNTER, S. W.; TRIVEDI, K. S. Analysis and implementation of software rejuvenation in cluster systems. In: **Proceedings of the 2001**

ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, ACM Press, USA, 2001c. p. 62-71.

VAIDYANATHAN, K.; GROSS, K. MSET Performance optimization for detection of software aging. Fast abstracts. In: **IEEE 14th International Symposium on Software Reliability Engineering**, USA, 2003, p. 17-20.

VAIDYANATHAN, K.; TRIVEDI, K. S. A comprehensive model for software rejuvenation. **IEEE Transactions on Dependable and Secure Computing**, v. 2, n. 2, p. 124-137, 2005.

VAN DER MEULEN, M. J. P.; BISHOP, P. G.; REVILLA, M. An exploration of software faults and failure behaviour in a large population of programs. In: **15th International Symposium on Software Reliability Engineering**, USA, 2004. p. 101-112.

VERÍSSIMO, P; LEMOS, R. DE. **Confiança no funcionamento: proposta para uma terminologia em português**. Relatório Técnico, Publicação conjunta INESC e LCMI/UFSC, 1989.

VERITAS. **VERITAS File System (VxFS) 3.3.2 - Patch 02**. Incident n. 31948, Document ID: 239339, VERITAS Software Corporation, 2001, Disponível em: <http://seer.support.veritas.com/docs/239339.htm> Acesso em: 23 nov. 2005.

WANG, X.; XU, J.; PHAM, C. H. An effective method to detect software memory leakage leveraged from neuroscience principles governing human memory behavior. In: **15th International Symposium on Software Reliability Engineering**, USA, 2004, p. 329-339.

XIE, W.; HONG, Y.; TRIVEDI, K. S. Software rejuvenation policies for cluster systems under varying workload. In: **10th IEEE Pacific International Symposium on Dependable Computing**, 2004. p. 122-129.

XIE, W.; HONG, Y.; TRIVEDI, K. S. Analysis of a two-level software rejuvenation policy. **Journal of Reliability Engineering and System Safety**, v. 87, ed. 1, p. 13-22, 2005.

YANG, J.; MELENOVSKY, M. IBM director: driving efficiencies in scale-out computing. **IDC Information and Data**, p. 5, mar. 2004.

YURCIK, W.; DOSS, D. Achieving fault-tolerant software with rejuvenation and reconfiguration. **IEEE Software**, vol. 18, ed. 4, p. 48-52, 2001.

APÊNDICE A

CÁLCULO DO TAMANHO DA AMOSTRA DO TEA

A.1. INTRODUÇÃO

Os procedimentos apresentados a seguir são usados para determinar o número de testes do TEA (segundo processo do método). O propósito da amostra é a estimativa da média e mediana dos tempos de falha/*pseudofalha* obtidos no TEA. Para outras estimativas, estes mesmos procedimentos podem ser adaptados como descrito nos capítulos 4 e 6 de Nelson (2004). Vale lembrar que estes procedimentos são válidos desde que satisfeitos os pressupostos listados na seção 5.4.3.1.

A.2. CÁLCULO DO NÚMERO DE TESTES DO TEA

O algoritmo a seguir realiza a transformação dos dados da amostra piloto utilizando $\log_{10}$ ou $\log_e$. A base do logaritmo depende do tipo de distribuição que será usada. Portanto, nas equações que envolvem a transformação de dados, o subscrito *t* será usado para representar a base do logaritmo. O valor de *t* é definido no item (d) do passo I do algoritmo. Antes da apresentação do algoritmo, a seguir são apresentadas as equações usadas no cálculo do tamanho da amostra do TEA:

$$\overline{x} = (n_1 x_1 + ... + n_j x_j) / np\,, \tag{A.1}$$

onde:

x_j é o valor do nível de estresse *j* transformado. No caso do método proposto, a utilização do IPL exige a transformação $x_j = \log_t(\textit{valor do estresse no nível } j)$;

n_j é o número de testes executados no nível de estresse *j*;

np é a quantidade total de testes da amostra piloto ($np = n_1 + n_1 ... + n_j$).

$$\bar{y}_j = (y_{1j} + y_{2j} + ... + y_{n_j j}) / n_j, \tag{A.2}$$

onde $y_{n_j j}$ é o $\log_t$ do *n*-ésimo tempo de falha/*pseudofalha* resultante de um teste no nível *j* de estresse.

$$s_j = \left\{ \frac{\left[(y_{1j} - \bar{y}_j)^2 + ... + (y_{n_j j} - \bar{y}_j)^2 \right]}{v_j} \right\}^{1/2}, \tag{A.3}$$

onde:

v_j é o número de graus de liberdade de s_j. Seu valor é $v_j = n_j - 1$;

s_j é o desvio padrão dos tempos obtidos em cada nível *j*.

$$s = \left[\frac{\left(v_1 s_1^2 + ... + v_j s_j^2 \right)}{v} \right]^{1/2}, \tag{A.4}$$

onde:

v é o número de graus de liberdade, calculado como $v = v_1 + ... + v_j$;

s é o estimador combinado[36] do $\log_t$ do desvio padrão (σ) em (A.5).

$$n_{TEA} = \left\{ 1 + (x_0 - \bar{x})^2 \left[\frac{np}{\sum (x - \bar{x})^2} \right] \right\} \left(\frac{z_{\alpha/2}\sigma}{\zeta} \right)^2 \tag{A.5}$$

onde:

x_0 é o $\log_t$ do valor do estresse para o qual se deseja obter as estimativas (nível de uso);

z é o valor tabulado da distribuição normal padrão para o nível de confiança α;

ζ é a precisão da estimativa, seu valor depende da distribuição de probabilidade dos tempos de falha (ver o passo "b" de II no algoritmo a seguir).

[36] Alguns autores (NELSON, 2004, p. 175) usam o termo estimador do desvio padrão baseado em replicação ou *pooled estimate*.

A.2.1. **Algoritmo**

I. Gerar amostra piloto:

a) Para a definição do tamanho desta amostra, o método baseia-se na sugestão de Nelson (2004, p. 236) em se considerar no mínimo 20 falhas (ver seção 5.4.2.1). O método considera planos com três níveis, portanto um tamanho sugerido para a amostra piloto seria de 21 testes, o que resultaria em 7 ensaios em cada nível de estresse (no plano tradicional).

b) Executar os testes de aceleração do envelhecimento (ver seção 5.4) para a obtenção dos tempos de falha/*pseudofalha* para a amostra piloto.

c) Avaliar o melhor ajuste entre as três distribuições consideradas (ver seção 5.5.1);

d) Para a Lognormal transformar os dados com o Log_{10} e para Weibull e Exponencial aplicar a transformação Log_e.

II. Cálculo do número de replicações (n_{TEA}) do TEA (ver equação A.5):

a) Definir o nível de confiança (1-α) desejado;

b) Definir a precisão ζ da estimativa. Para a média considera-se $\zeta = r$ e para a mediana $\zeta = \log_t(r)$, onde em ambos os casos *r* é a precisão que se deseja do estimador em relação ao verdadeiro valor, sendo o semi-intervalo usado para obter o intervalo de confiança.

Para $\zeta = r$ seu significado é o mesmo de E_0 em (5.4) (ver seção 5.3.1). Seu valor está na mesma unidade da média e *n* é calculado para que a estimativa da média esteja no intervalo de ±*r* do verdadeiro valor.

Para $\zeta = \log_t(r)$, tem-se *r* = (1 + *m*), onde *m* x 100% é o erro tolerado (em termos percentuais) em relação à verdadeira mediana ($\tilde{x}$) dos tempos de falha/*pseudofalha*. Por exemplo, *m* = 0,20 significa que *n* é calculado para que o verdadeiro valor de $\tilde{x}$ esteja no intervalo de $\frac{\tilde{x}}{r}$ até $\tilde{x} \cdot r$;

c) Com os dados (transformados) da amostra piloto, obter n_{TEA} computando as equações (A.1) até (A.5).

Se o valor de n_{TEA} for menor ou igual ao tamanho da amostra piloto sugerida (*np*=21), então os dados da amostra piloto são suficientes para os demais procedimentos do TEA. Em caso contrário, (21 – *np*) testes devem ser realizados para complementar os dados da amostra piloto.

APÊNDICE B

RESULTADOS DA ANÁLISE DA AMOSTRA PILOTO DO TEA

Tabela B.1: Valores da variável resposta do DOE para cada replicação dos tratamentos

	T1	T2	T3	T4	T5	T6	T7	T8
R1	24000	24000	24000	24000	49020	47992	68924	65936
R2	24000	24000	24000	24000	51224	49568	70316	70748
R3	24000	24000	24000	24000	47060	54672	68196	68008
R4	24000	24000	24000	24000	50760	52340	68644	70120
R5	24000	24000	24000	24000	54416	51216	64588	66704
R6	24000	24000	24000	24000	51516	52136	75052	70376
R7	24000	24000	24000	24000	48208	49348	65048	62192
R8	24000	24000	24000	24000	48908	47972	72832	66224
R9	24000	24000	24000	24000	50804	54568	75160	68256
R10	24000	24000	24000	24000	53412	58352	60776	67964
R11	24000	24000	24000	24000	48828	50476	71908	65968
R12	24000	24000	24000	24000	48152	52612	67448	68488
R13	24000	24000	24000	24000	49364	50496	62044	73240
R14	24000	24000	24000	24000	49416	51468	69808	70732
R15	24000	24000	24000	24000	52792	51184	70344	68956
R16	24000	24000	24000	24000	49096	49592	60812	67748
R17	24000	24000	24000	24000	49748	50028	69948	62812
R18	24000	24000	24000	24000	46256	50860	70972	66196
R19	24000	24000	24000	24000	47772	52940	67816	71184
R20	24000	24000	24000	24000	50656	51940	70008	66728
R21	24000	24000	24000	24000	51384	51848	66680	64616

Tabela B.2: Parâmetros dos modelos logarítmicos ajustados aos dados de degradação do TEA piloto

Replicação	Nível de estresse	Parâmetros		R^2
		a	*b*	
1	S1	63203,85	-189179,43	0,99
2	S1	66393,65	-208111,78	0,99
3	S1	66581,10	-210652,98	0,99
4	S1	64888,68	-212607,15	0,99
5	S1	65438,55	-204847,32	0,99
6	S1	65902,89	-206872,59	0,99
7	S1	62764,51	-192696,11	0,99
1	S2	72856,63	-133629,54	0,99
2	S2	68435,58	-102810,95	0,99
3	S2	70910,52	-119604,91	0,99
4	S2	79250,94	-178249,67	0,99
5	S2	71846,93	-123421,95	0,99
6	S2	78013,10	-166303,02	0,99
7	S2	72702,40	-134223,53	0,99

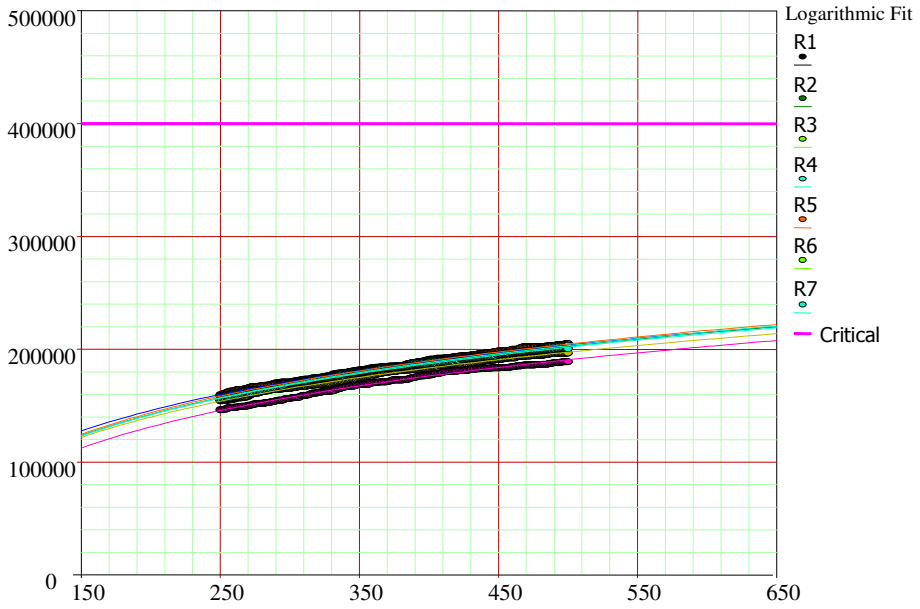

Figura B.1: Modelos logarítmicos ajustados aos dados de degradação do nível de estresse S1

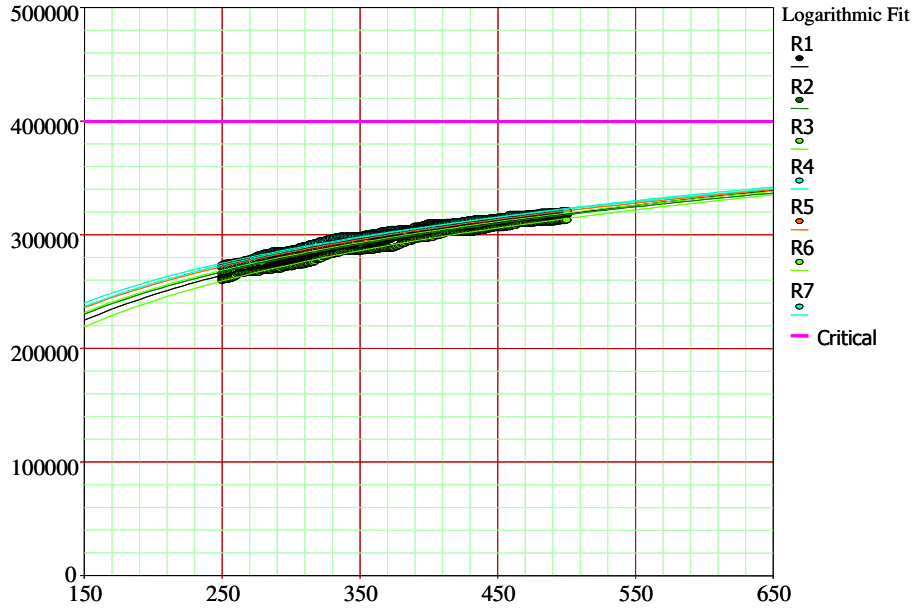

Figura B.2: Modelos logarítmicos ajustados aos dados de degradação do nível de estresse S2

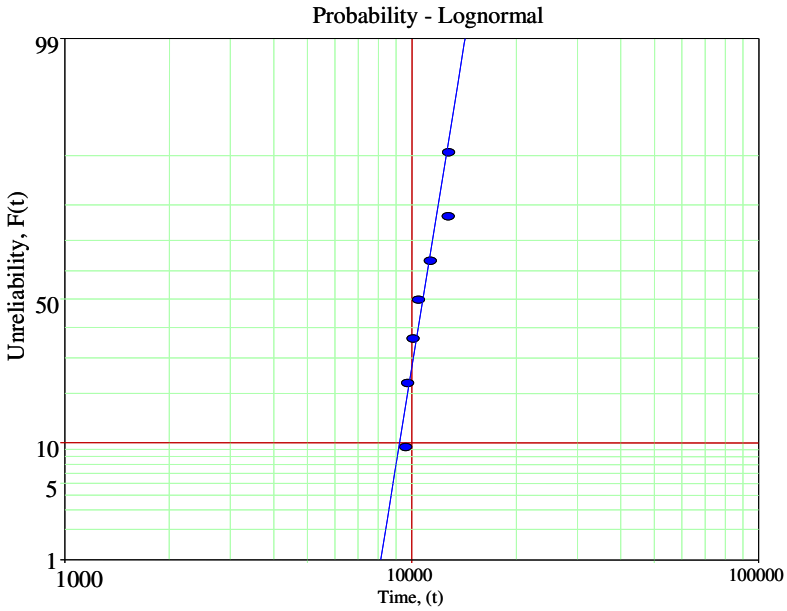

Figura B.3: Gráfico de probabilidade Lognormal com ajuste por MLE para o nível S1

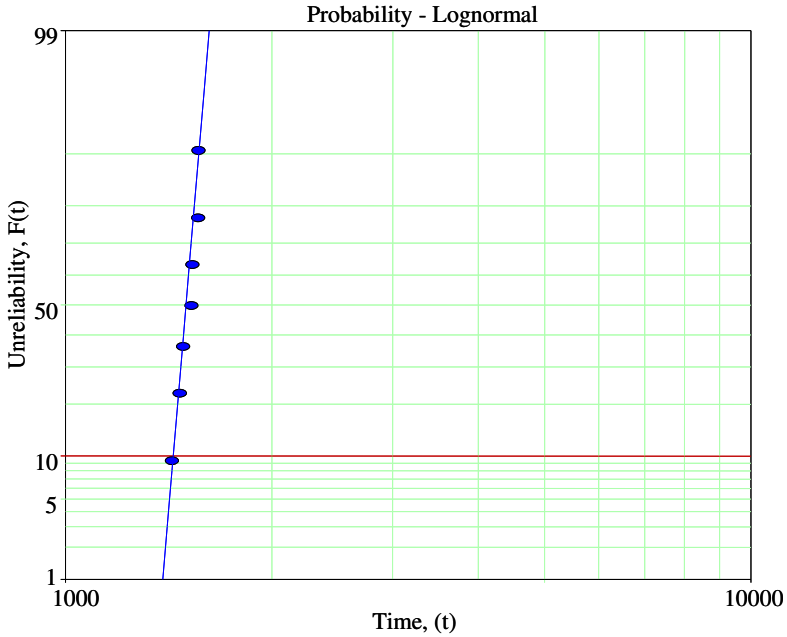

Figura B.4: Gráfico de probabilidade Lognormal com ajuste por MLE para o nível S2

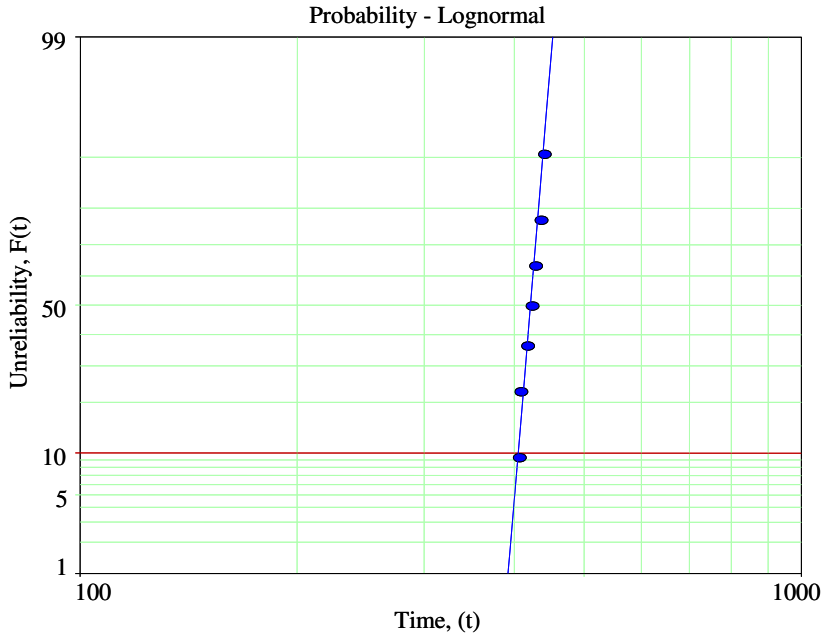

Figura B.5: Gráfico de probabilidade Lognormal com ajuste por MLE para o nível S3

APÊNDICE C

CARATERIZAÇÃO DA CARGA DE TRABALHO

Como descrito no primeiro processo do método proposto (ver figura 5.6), uma de suas entradas é a carga de trabalho caracterizada para o nível de uso do sistema analisado. A caracterização descrita neste apêndice objetivou encontrar o tempo médio entre chegadas (TEC), de requisições HTTP para páginas de conteúdo dinâmico, destinada a um servidor web real. Esta análise considerou apenas páginas dinâmicas, visto ser este o fator mais significativo dentre aqueles avaliados e que compõem o fator de envelhecimento (F_E) (ver seção 6.4.1.2). A análise foi conduzida com base em dados de tráfego HTTP coletados do mesmo ambiente de ISP descrito na seção 6.4.1.1.

Primeiramente, foram selecionados os dez endereços (URL) de páginas dinâmicas mais acessados do ISP. A partir destes endereços, obteve-se a distribuição dos tempos entre chegadas (TEC) de requisições HTTP destinadas a estas páginas. A seleção dos endereços mais acessados, bem como a obtenção dos tempos entre chegadas, teve como base os dados de três meses de registros do *log* do servidor Apache do ISP, resultando em uma amostra de 16.946 acessos. A partir desta amostra, foram realizados testes de aderência para avaliar o ajuste de distribuições de probabilidades aos dados, de forma a obter o modelo que melhor representasse o comportamento dos tempos entre chegadas. Os cinco melhores ajustes estão listados na tabela C.1. Estes testes foram realizados com o auxílio do software @Risk BestFit (ver seção 5.5.1).

Tabela C.1: Resultado do teste K-S contra a amostra de TEC do ISP

Ranking	Modelo	Valor *p*	Distância K-S
1	Lognormal	0,5983	0,006497
2	InvGauss	0,4286	0,007301
3	Pearson	0,0129	0,012740
4	LogLogistic	0,0000	0,017280
5	Normal	0,0000	0,135300

Como se pode observar, a Lognormal foi a distribuição de probabilidades que melhor se ajustou aos dados da amostra. A figura C.1 apresenta a PDF desta distribuição, a qual foi estimada a partir dos dados da amostra.

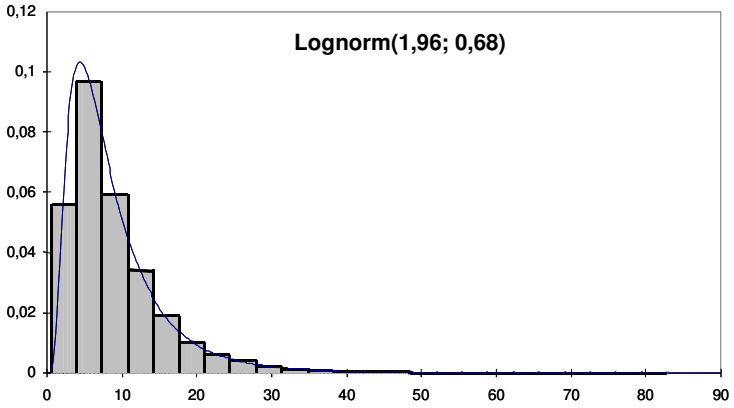

Figura C.1: Distribuição observada e modelo teórico para a amostra de TEC.

Os valores no eixo x representam os tempos entre chegadas em segundos. O valor médio e o desvio padrão da amostra foram 9 seg. e 7 seg., respectivamente. Desta forma, ficou caracterizado que o sistema em questão apresentou uma taxa média de requisições HTTP, para páginas com conteúdo dinâmico, de 0,11 requisição/segundo.

APÊNDICE D

RESULTADOS DO TEA PARA O CENÁRIO D_F = 100 MEGABYTES

7.5. ETAPA DE PLANEJAMENTO E EXECUÇÃO DO TEA

Tabela D.1: Sumário dos elementos de forma do TEA

	Nível de uso (NU)	Sl (2 x NU)	S2 (3 x NU)	S3 (4 x NU)
Tamanho de página (kB)	196	392	588	784
Taxa de requisições do F_E (req/sec)	-	29	19	14
Número de requisições (D_t)	-	50.000	50.000	50.000
Número de inspeções (t)	-	500	500	500
Limiar de falha (D_f) (megabytes)	-	100	100	100

TTF(NU)	TTF (S1)	TTF (S2)	TTF (S3)
329	84	34	20
331	86	36	21
334	88	37	22
344	93	38	23
351	95	38	23
357	95	39	23
359	97	40	24

Quadro D.1: Amostra de tempos de falha do TEA para os níveis de estresse e o nível de uso

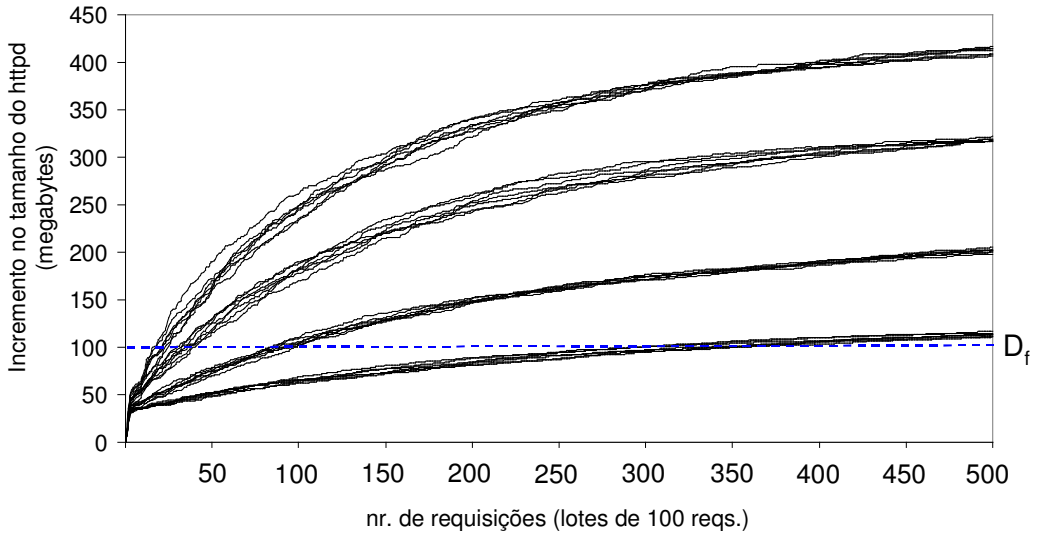

Figura D.1: Caminhos de degradação do TEA para os níveis de estresse e nível de uso

Tabela D.2: Resultado do teste de aderência

Nível de Estresse	Modelo	GOF		Ranking
		Lk	*ρ* (%)	
S1	Weibull	-20,5086	96,75	1º
	Lognormal	-20,8627	95,98	2º
	Exponencial	-38,5870	-74,38	3º
S2	Weibull	-13,9211	99,31	1º
	Lognormal	-14,3368	97,60	2º
	Exponencial	-32,3570	-74,53	3º
S3	Weibull	-11,2420	97,65	1º
	Lognormal	-11,8037	95,33	2º
	Exponencial	-28,7276	-73,96	3º

Como se pode observar na tabela D.2, o melhor ajuste se deu para a distribuição de probabilidades Weibull, diferente do cenário de D_f=400 em que a melhor aderência foi da distribuição Lognormal. Neste caso, de acordo com o algoritmo descrito no apêndice A, a transformação dos dados da amostra deve ser realizada com o logaritmo natural (Log_e), cujo resultado está apresentado no quadro D.2.

TTF (S1)	TTF (S2)	TTF (S3)
1,9243	1,5315	1,3010
1,9345	1,5563	1,3222
1,9445	1,5682	1,3424
1,9685	1,5798	1,3617
1,9777	1,5798	1,3617
1,9777	1,5911	1,3617
1,9868	1,6021	1,3802

Quadro D.2: Resultado do Log_e da amostra piloto do TEA para S1, S2 e S3

A partir dos dados transformados, o número de replicações do TEA (n_{TEA}) foi calculado executando os procedimentos descritos no item II da seção A.2.1, os quais estão sumarizados a seguir:

(A.1) $\bar{x} = 2{,}752326$

(A.2) $\bar{y}_1 = 1{,}9591$, $\bar{y}_2 = 1{,}5727$, $\bar{y}_3 = 1{,}3473$

(A.3) $s_j = 0{,}0244$, $s_j = 0{,}0234$, $s_j = 0{,}0274$

(A.4) $s = 0{,}0251$

(A.5) $$n_{TEA} = \left\{1 + (x_0 - \bar{x})^2 \left[\frac{np}{\sum (x - \bar{x})^2}\right]\right\}\left(\frac{z_{\alpha/2}\sigma}{\zeta}\right)^2 = 11{,}21093,$$

onde:

$$x_0 = \log_{10}(196) = 2{,}2923;$$
$$\alpha = 0{,}95,\ z_{\alpha/2} = 1{,}96;$$
$$r = (1+m),\ m{=}0{,}10;$$
$$\zeta = \log_{10}(r) = \log_{10}(1{,}10) = 0{,}0414;$$
$$\sigma \approx s$$

Como resultado, o n_{TEA} calculado foi de 11,21. Tendo em vista a adoção de um plano tradicional, cuja alocação é idêntica nos três níveis de estresse, este valor poderia ser arredondado para 12, resultando em 4 replicações para cada nível de estresse. Contudo, a amostra piloto foi superior ao tamanho de amostra calculado e, portanto, decidiu-se trabalhar com a própria amostra piloto.

Tabela D.3: Plano tradicional adotado para a execução do TEA

Nível de estresse		Alocação	
Condição	TMP (F_E) (*kilobytes*)	Proporção π_i	Número de testes n_i
Uso	196	-	-
N_b	392	1/3	7
N_i	588	1/3	7
N_a	784	1/3	7

7.6. MODELAGEM DO RELACIONAMENTO ESTRESSE-ENVELHECIMENTO ACELERADO

Como a amostra piloto foi suficiente para representar o número de replicações do TEA, a distribuição Weibull será usada em conjunto com o relacionamento IPL para a modelagem do relacionamento estresse-envelhecimento acelerado.

De acordo com Meeker e Escobar (1998, p. 86), a distribuição Weibull pode adotar a mesma parametrização definida em (4.5) (ver seção 4.2.5.2), devido o seu relacionamento com a distribuição de menor valor extremo (MEEKER; ESCOBAR, 1998, p. 83; FREITAS; COLOSIMO, 1997, p. 143). Neste caso, $\sigma = 1/\beta$ é o parâmetro de escala e $\mu = \log(\eta)$ o parâmetro de localização. Portanto, a suposição de se ter o mesmo parâmetro de escala, em todos os níveis de estresse, deve ser avaliada com base nos valores estimados para o parâmetro β da tabela D.4. O valor de $\hat{\beta}$ deve estar dentro do mesmo IC estimado para todos

os níveis. Este valor foi ajustado utilizando o software ALTA Pro 6, o que resultou em um $\hat{\beta}$ = 18,94.

Tabela D.4: Parâmetros estimados para o modelo Weibull em cada nível de estresse

Estresse	Parâmetros	Estimativa (MLE)	IC (90%)	
			LI	LS
S1	β_1	24,0889	14,3941	40,3134
	η_1	93,3175	90,8149	95,8891
S2	β_2	25,0000	15,2671	40,9377
	η_2	38,2697	37,2791	39,2865
S3	β_3	22,0825	13,3316	36,5775
	η_3	22,8595	22,1925	23,5465

Tabela D.5: Parâmetros estimados para o modelo IPL-Weibull

Parâmetro	Estimativa (MLE)	IC (90%)	
		LI	LS
K	5,7869E-8	3,7257E-8	8,9885E-8
N	2,0340	1,9646	2,1034
β	18,9434	13,5270	26,5286

7.7. ESTIMAÇÃO DA DISTRIBUIÇÃO DE VIDA PARA O NÍVEL DE USO

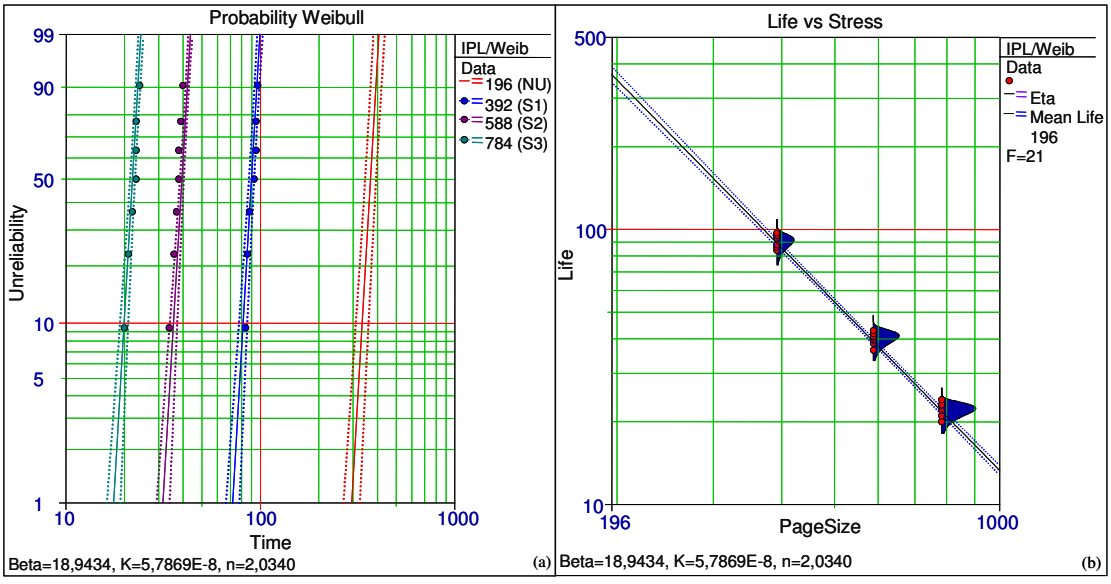

Figura D.2: (a) Gráfico múltiplo de probabilidade Weibull com ajuste do modelo IPL-Weibull; (b) Comportamento da vida média (TTF) estimada em função do estresse

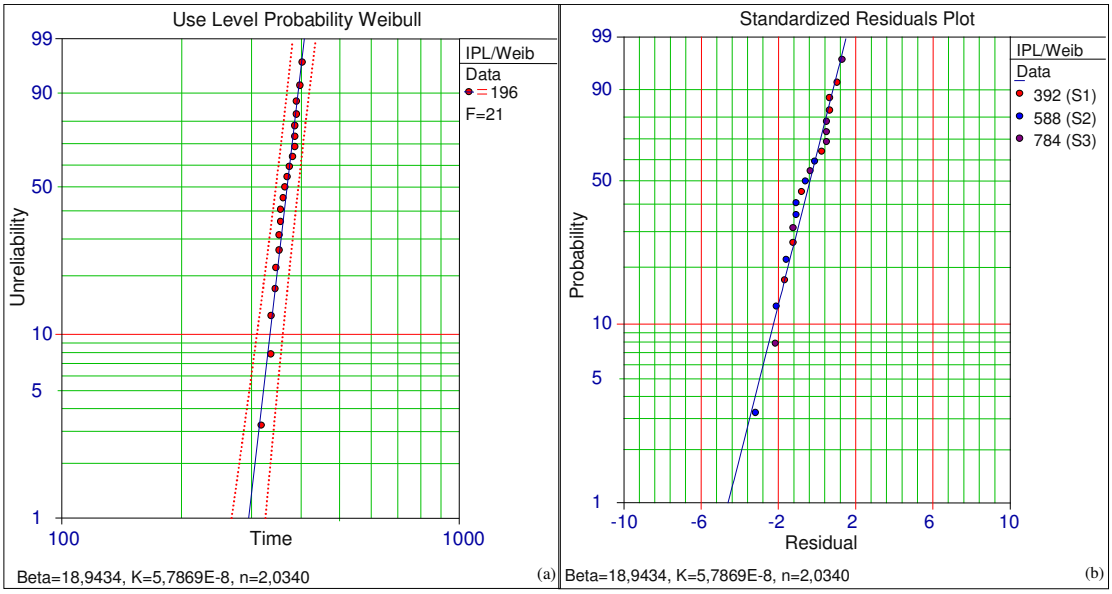

Figura D.3: (a) Gráfico de probabilidade Weibull para o nível de uso; (b) Gráfico de probabilidade normal de resíduos do modelo IPL-Weibull

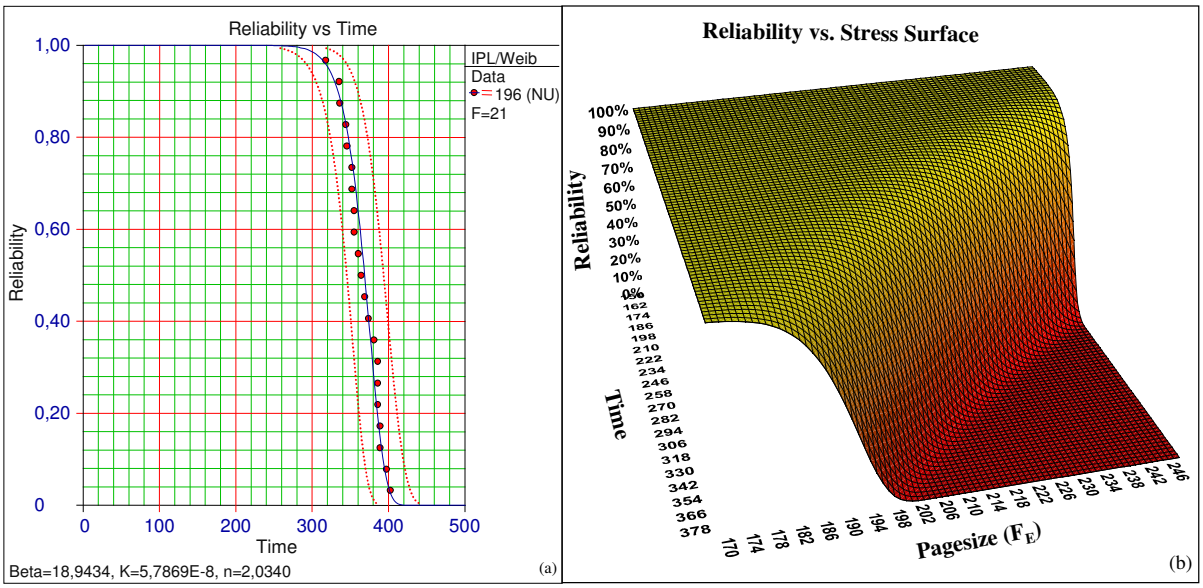

Figura D.4: (a) Função confiabilidade para o nível de uso (TMP=196); (b) Função confiabilidade contra o tempo e o F_E (tamanho de página dinâmica)